Contributions to Pattern Discovery and Formal Concept Analysis

Mehdi Kaytoue

UNIVERSITÉ DE LYON

École doctorale InfoMath de Lyon

Contributions to Pattern Discovery and Formal Concept Analysis

THÈSE

présentée et soutenue publiquement le 12 février 2020

pour l'obtention d'une

Habilitation de l'Institut National des Sciences Appliquées de Lyon et l'Université Claude Bernard LYON I

(mention informatique)

par

Mehdi Kaytoue

Composition du jury

Rapporteurs :	Dr. Karell Bertet	Maître de conférences, Université de la Rochelle
	Dr. Florent Masseglia	Directeur de recherche, INRIA
	Pr. Christel Vrain	Professeur, Université d'Orléans
Examinateurs :	Pr. Michael Berthold	Professeur, Universität Konstanz (Allemagne)
	Pr. Angela Bonifati	Professeur, Université Claude Bernard Lyon 1
	Pr. Jean-François Boulicaut	Professeur, INSA Lyon
	Pr. Johannes Fürnkranz	Professeur, Universität Linz (Autriche)
	Dr. Amedeo Napoli	Directeur de recherche, CNRS

Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS) — UMR 5205

Mis en page avec la classe thesul.

Acknowledgements

Je remercie en premier lieu les membres du jury de cette habilitation. Leur renommée, leur expertise et l'intérêt qu'ils ont suscité sur ces travaux sont honneur et source de fierté. L'organisation de la soutenance ne fut pas tâche facile, les délais étaient courts et votre séjour éphémère. Malgré tout, la soutenance s'est déroulée dans un cadre académique idéal.

Je remercie ensuite les principaux artisans des travaux de recherche présentés dans ce manuscrit, doctorants, ingénieur et post-doctorant que j'ai eu la chance d'encadrer. En particulier, Olivier, Guillaume, Aimene, Romain, Victor et Pierre : Vos qualités humaines, vos idées, vos réalisations, et surtout la manière dont vous vous êtes accaparés les différentes idées ou problématiques sont remarquables. La présentation de cette habilitation n'était pas possible sans cela. Je suis honoré de continuer de collaborer avec nombre d'entre vous.

Je remercie alors l'ensemble des collègues avec qui j'ai collaboré ces dix dernières années. Ils sont trop nombreux pour que je les nomme ici un à un. C'est à travers votre contact, et la plupart du temps votre passion pour la recherche, que j'ai le plus appris.

Je remercie aussi mes collègues d'INFOLOGIC, entreprise avec des problématiques passionnantes exposées par des passionnés, qui me permet de continuer sereinement mon activité de recherche et d'avoir préparé cette habilitation.

Il ne faut bien sûr pas oublier les étudiants de l'INSA. Votre contact a toujours été enrichissant (même en retard ou du dernier rang) et je ne garde que de bons souvenirs.

Il y a des mentors, à chaque étape cruciale d'un parcours, qui vous inspirent tout particulièrement, vous fascinent, vous amènent à (vous) remettre en question, et tout simplement, auxquels vous êtes attachés bien plus que par un objet professionnel commun. Depuis le début, il y a Amedeo. Il y a ensuite eu Sergei, Chedy, Jean-François, André. Merci.

Merci à ma famille, mon épouse et mes deux merveilleuses filles, je ne sais toujours pas comment vous arrivez à me supporter ! Je vous aime.

Contents

Introduction **5**

1 Introducing Knowledge Discovery in Databases 5

2 Formal Concept Analysis 7

3 Subgroup Discovery and Algorithms 13

4 Knowledge Discovery in Practice 17

5 Outline 23

Part I Formal Concept Analysis **25**

Chapter 1
Biclustering

1.1 Introduction 27

1.2 Problem Settings 29

1.3 Triadic Concept Analysis 30

1.4 Biclusters of Similar Values in Triadic Concept Analysis 31

1.5 Extracting Biclusters of Similar Values For a Given θ 37

1.6 Conclusion 40

Chapter 2
Database Dependency Discovery

2.1 Introduction 41

2.2 Functional and Degenerated Multivalued Dependencies 42

2.3 Characterizing Functional Dependencies with FCA 44

2.4 Characterizing FDs with Pattern Structures 46

2.5 Characterizing DMVDs with Pattern Structures 49

2.6 Conclusion 51

Chapter 3
Numerical Pattern Mining

3.1 Introduction . . . 53
3.2 Interval Patterns . . . 54
3.3 Convex Polygon Patterns . . . 55
3.4 Algorithms . . . 57
3.5 Conclusion . . . 60

Part II Pattern Mining and Subgroup Discovery 63

Chapter 4
Pattern Discovery with Monte Carlo Tree Search

4.1 Introduction . . . 65
4.2 Pattern Set Discovery . . . 66
4.3 Monte Carlo Tree Search . . . 69
4.4 Pattern Set Discovery with MCTS . . . 72
4.5 Conclusion . . . 79

Chapter 5
Anytime Subgroup Discovery in Numerical Domains with Guarantees

5.1 Introduction . . . 83
5.2 Preliminaries . . . 84
5.3 Problem Statement . . . 86
5.4 Anytime Interval Pattern Mining . . . 86
5.5 Anytime Interval Pattern Mining with Guarantees . . . 88
5.6 Conclusion . . . 91

Part III Video Game Analytics 93

Chapter 6
Discovering Opening Strategies in RTS Games

6.1 Introduction . . . 95
6.2 Preliminaries . . . 96
6.3 The Problem of Strategy Elicitation . . . 97

6.4 Balanced Patterns in Non-mirror Databases 98
6.5 Balanced Patterns in Mirror Databases 99
6.6 Algorithms 100
6.7 Experiments 100
6.8 Related Work 104
6.9 Conclusion 104

Chapter 7
Revealing the Identity of Anonymous Players

7.1 Introduction 105
7.2 Problem Settings 106
7.3 Mining a Confusion Matrix with FCA 108
7.4 Evaluation 111
7.5 Experiments 115
7.6 Related Work 118
7.7 Conclusion 120

Conclusion **121**

Bibliography **125**

Introduction

Starting this manuscript by introducing Knowledge Discovery in Databases (KDD) is nowadays old-fashioned and the reader may even find it boring if she/he used to work in the field for many years. It was indeed a *buzz-word* twenty years ago, and a myriad of articles, PhD theses, and probably "Habilitations" too, were massively introducing and then enhancing the KDD process. After KDD, the terms "big data", then "data science" got more attractive, to end up today with the general term of "Artificial Intelligence", covering of course many other problems and gathering governments, academics and companies.

However, KDD is what reflect the better what domain experts need when attempting to understand a phenomena from collected data and highlight the concepts that take part in it or even the process that generates the data itself. It is what drove the majority of my research activities, through projects and collaborations with academics and industries. It is also an exciting field to teach, especially through practical courses, where students get surprised in what they can discover in a dataset.

As a matter of fact, this manuscript will first give an overview of our main contributions over the last ten years on theoretical, algorithmic and methodological aspects of KDD, but also our experience with knowledge discovery in practice in different application domains. We will then dive in the manuscript on a selection of contributions (generally on which others were built), before concluding with our perspectives of research.

1 Introducing Knowledge Discovery in Databases

We are living in a world of data. Huge volumes of data –web documents, user information, sensor data– are available, sometimes without any intended usage but for having legal archives. Large volumes of biological data are available –genome, transcriptome, proteome, metabolome, 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 a very large amount of data. Accordingly, a challenging question arose in the 90's: "Can we make (large) data speak?". This question still needs today theoretical foundations, algorithms, methodologies, and beyond all, a lot of practice, to be answered.

1.1 The KDD process

Knowledge discovery in databases (KDD) is the process of finding non-trivial, potentially useful, significant and reusable information in data [59, 58]. 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 several steps [58, 59, 55].

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.). Nowadays, sensors producing data are everywhere.

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.

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, clusters or classes, etc.). It consists in pattern discovery or deriving/designing a model from the data. Pattern-mining is actually the focus of most of our research and we detail it in the subsection.

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. This step is gaining a lot of attention these last years, coined with the term "Interpretability". Here, data-mining techniques can also be used to give hypotheses on the reasons a black-box model gives its predictions.

1.2 Eliciting Hypotheses from Data

In our research, we were mainly interested in *pattern discovery*, or the task to discover interesting regularities in the data, which translate into hypotheses for the domain expert. These hypotheses, when interpreted and validated by the expert, can (i) be actionable, that is, result in an immediate decision, or (ii) be incorporated in a knowledge base, e.g., to be reused and guide other explorations of the data through KDD. Our research on pattern discovery can be divided into three major axes.

Data and Pattern Formalization A pattern mining task consists in discovering small and interesting parts of the data having nice properties, such as, for a simple example, the fact of being frequent and discriminant for a particular part of the dataset. However, the type of patterns that can be discovered in a dataset depends of its structure. One can discover frequent item sets in a binary matrix, but frequent sub-graphs in a collection of graphs. As such, the first step is to formalize the data set and the patterns to discover within. For that, we chose to anchor our work in order theory, and especially Formal Concept Analysis. We focus in this axis of research on defining and understanding the mathematical objects that pattern mining manipulates.

Mining Algorithms After properly formalizing that data set and the patterns we seek, the next question is to solve the search problem with efficient algorithms, bringing the expert with few but new and possibly useful hypotheses. Even dealing with the most simple type of patterns brings the end-user with zillions of patterns, that is, useless patterns. Defining interestingness measures, and handling them efficiently during the search to output a small, non redundant, yet diverse, set of patterns, requires smart algorithmic strategies. Many efficient exhaustive enumeration techniques have been proposed between 1990 and 2010, and can serve as a basis for designing new "non-exhaustive" search strategies, with some guarantees. We started to investigate some exhaustive search techniques, before focusing on anytime algorithms: the more the budget is given, the best is the result, until, in theory, being the same than the one an exhaustive search would produce.

Methodologies for Applications Formalizing a pattern mining task and solving it with an algorithm is one thing. Discovering actionable patterns from a dataset is another (long) story. Actually, our need of anytime algorithms and particular pattern quality measures rose from applications. For

example, we worked on eliciting links between molecules (described by thousands of categorical and numerical attributes) and odors given by scent experts (multi-label settings). The dataset dimensionality, the need of precise patterns (forbidding the use of numerical attribute discretization, thus implying a huge search space), the skewed label distribution, the possible correlations between these labels, etc. led us to rethink some search strategies and algorithmic paradigms. In the end, what appeared to be a simple application problem in neuroscience brought us in a four year collaboration before ending up with results published in a impactful application domain journal (PLOS Computational Biology in this case).

The following thus sum up our works in three sections. Of course, any contribution mentioned in this document is the product of exciting and fruitful collaborations with other researchers (both "seniors" and "juniors") and both industrial and academic partners and we make it precise hereafter.

2 Formal Concept Analysis

Concepts are necessary for expressing human knowledge, hence the KDD process should benefit from a comprehensive formalization of concepts [166]. Formal Concept Analysis (FCA) [64] 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 conceptual knowledge discovery in databases, and revealed itself to be helpful for applications in information and knowledge processing including visualization, knowledge management, and, for our concerns, pattern mining.

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 [165]. It rapidly grows into a research field leading to a seminal book [64] 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 [166, 156, 100, 136, 135]. A website dedicated to FCA is maintenend by Uta Priss[1].

From a more personal point of view, FCA represents my major research activity. I am part of the steering committee of the International Conference on Formal Concept Analysis (ICFCA) since I co-chaired the 2014's edition [71].

```
Cynthia Vera Glodeanu, Mehdi Kaytoue, Christian Sacarea:
12th Int. Conf. on Formal Concept Analysis (ICFCA 2014).
Lecture Notes in Computer Science 8478, Springer.
```

Before illustrating my main contributions, let us briefly introduce FCA.

2.1 A Short Introduction to FCA

We now make precise a few general notions in FCA necessary for the understanding of this document.

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

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

	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	×		×		×	×

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

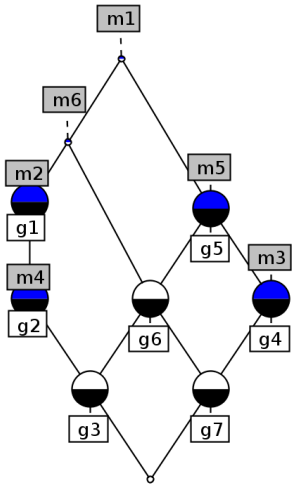


Figure 2: Concept lattice raised from Table 1

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 maximal set of objects A and a maximal 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).

Example 1 *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. 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.*

Concept forming 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: $A' = \{m \in M \mid gIm \ \forall g \in A\}$. In a dual way, for a set of attributes $B \subseteq M$, we define the set of objects that have all attributes from B as: $B' = \{g \in G \mid gIm \ \forall m \in B\}$. 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 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.

Example 2 *We have $\{g_1, g_2\}' = \{m_1, m_2, m_6\}$ and $\{m_1, m_2, m_6\}' = \{g_1, g_2, g_3\}$. 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: $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_6\}) \leq (\{g_1, g_2, g_3, g_6, g_7\}, \{m_1, m_6\})$.*

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.

Example 3 *Figure 2* [2] *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.* $\leq$.

Implication. An implication of a formal context (G, M, I) is denoted by $X \rightarrow Y$, $X, Y \subseteq M$ and means that all objects from G having the attributes in X also have the attributes in Y, i.e. $X' \subseteq Y'$. Implications obey the Amstrong rules (reflexivity, augmentation, transitivity). A minimal subset of implications (in sense of its cardinality) from which all implications can be deduced with Amstrong rules is called the Duquenne-Guigues basis [74].

Example 4 $m_2 \rightarrow m_1$ *is an implication as* $m_2' \subseteq m_1'$, *that is* $G \subseteq \{g_1, g_2, g_3\}$. *Implications can be easily read on the diagram representation when using reduced labeling thanks to inheritance:* $m_2 \rightarrow m_6$ *but also* $m_3 \rightarrow m_5$.

Pattern structure To handle non binary data directly, the latter can be formalized as a pattern structure given some conditions. 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 $A = d^{\square}$. 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. As for formal contexts, implications can be defined. For $c, d \in D$, the pattern implication $c \rightarrow d$ holds if $c^{\square} \subseteq d^{\square}$, i.e. the pattern d occurs in an object description if the pattern c does. Similarly, for $A, B \subseteq G$, the object implication $A \rightarrow B$ holds if $A^{\square} \sqsubseteq B^{\square}$, meaning that all patterns that occur in all objects from the set A also occur in all objects in the set B [62].

Concept enumeration algorithms There exists several algorithms to compute concepts (sometimes just their extents or their intents), with or without their covering relation [99]. In data-mining, we are mainly interested in the set of intents with the cardinality of their extent, that corresponds to the *frequent closed itemsets* in the pattern mining domain. One algorithm that we used intensively is *CloseByOne*. Its power lies in two facts (i) it is conceptually the same algorithm than LCMv2 (the most popular and efficient algorithm for mining closed itemsets), and (ii) it can be used for processing pattern structures with slight modifications (computing the intersection).

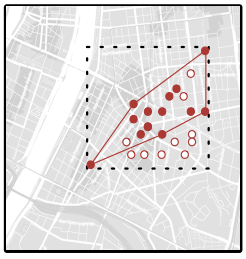
[2] Drawn with ConExp`http://conexp.sourceforge.net/`

Contributions to Formal Concept Analysis During my PhD (2007-2011), I investigated how FCA could help in processing numerical data for knowledge discovery and pattern mining purposes. For that, I collaborated with Pr. Sergei O. Kuznsetov, unburrowing an old work of his: *Pattern Structures* and its ability to consider data where objects are not described by itemsets (that is, binary data), but with *complex descriptions* [62]. Pattern structures, although not democratized, are simple and bring a comfort in formalizing a dataset and the patterns we are looking for: itemsets, hyper-rectangles, functional dependencies (and their generalizations), biclusters, among others, and to some extent sequential and graph data.

Consequently, this section is dedicated to our works on FCA during the last ten years, for its formalization and representation capabilities.

- We formalized strong links between numerical biclustering and n-set pattern mining (during my post doctoral research).
- We formally defined in a unifying framework several types of data dependencies (through a long term collaboration with Amedeo Napoli, Víctor Codocedo and Jaume Baixeries)
- We proposed a new numerical pattern domain: convex polygons (Aimene Belfodil's PhD).
- We proposed to extend the pattern structure models to pattern multi-structures and understand what kind of pattern languages fit into pattern structures, pattern multi-structures and pattern setups (Aimene Belfodil's PhD and collaboration with Sergei O. Kuznetsov). A pattern setup is the less restrictive data representation in which patterns can be found.

2.2 Patterns in Numerical Data



Pattern mining is an important task in AI for eliciting hypotheses from the data. When it comes to spatial data, the geo-coordinates are often considered independently as two different attributes. Consequently, rectangular shapes are searched for. Such an arbitrary form is not able to capture interesting regions in general. The problem with rectangular shapes can be observed on the right hand side figure. Each object gives a POI (Point Of Interest) of a given type (Hotel, Restaurant, University, ...) and position. An interesting pattern is understood as a geographical area for which there is a sufficient number of points, high density, and a high proportion of objects of the same type. The candidate areas could have any shape. Rectangles, as being the products of intervals, have edges parallel to the plane axes: they may enclose both dense and sparse regions. Arbitrary polygons stick too much to the data and are hard to interpret. We consider convex polygons, a good trade-off for capturing high density areas. Our contribution is threefold: (i) We formally introduce such patterns in Formal Concept Analysis, (ii) we give all the basic bricks for mining convex polygons with exhaustive search and pattern sampling, and (iii) we design several algorithms, which we compare experimentally. To the best of our knowledge, it is the first attempt to formally define this new type of pattern with FCA: the conjunction of hyper-plans defined over multiple numerical attributes.

This work actually extends our first proposition to consider numerical pattern as hyper-rectangles with FCA [91, 90]:

```
Aimene Belfodil, Sergei O. Kuznetsov, Céline Robardet, Mehdi Kaytoue:
Mining Convex Polygon Patterns with Formal Concept Analysis.
International Joint Conference on Artificial Intelligence (IJCAI 2017)
```

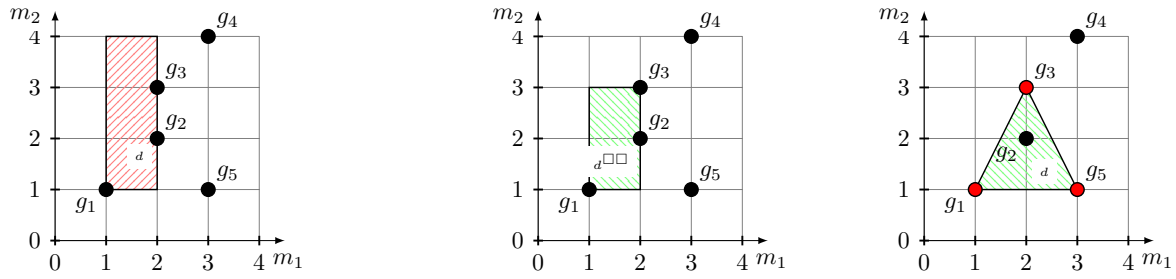


Figure 3: Non-closed (left), closed (middle) interval pattern and convex polygon (right) .

2.3 Mining Closed Sets Using Implications

FCA provides a mathematical tool to analyze and discover concepts in Boolean datasets (i.e. Formal contexts). It does also provide a tool to analyze complex attributes by transforming them to Boolean ones (i.e. items) thanks to *conceptual scaling*. For instance, a numerical attribute whose values are $\{1, 2, 3\}$ can be transformed to the set of items $\{\leq 1, \leq 2, \leq 3, \geq 3, \geq 2, \geq 1\}$ thanks to interordinal scaling. Such transformations allow us to use standard algorithms like CloseByOne to look for concepts in complex datasets by leveraging a closure operator. However, these standard algorithms do not use the relationships between attributes to enumerate the concepts as for example the fact that ≤ 1 implies ≤ 2 and so on. For such, they can perform additional closure computations which substantially degrade their performance. We propose in this work a generic algorithm CbOi (CloseByOne using implications) to enumerate concepts in a formal context using the inherent implications between items provided as an input. We show that using the implications between items can reduce significantly the number of closure computations and hence the time effort spent to enumerate the whole set of concepts.

```
Aimene Belfodil, Adnene Belfodil, Mehdi Kaytoue:
Mining Formal Concepts Using Implications Between Items.
International Conference on Formal Concept Analysis (ICFCA 2019)
```

2.4 Biclustering

Biclustering numerical data became a popular data mining task at the beginning of 2000's, especially for gene expression data analysis and personal recommendations. A bicluster reflects a strong association between a subset of objects and a subset of attributes in a numerical object/attribute data-table. So called biclusters of similar values can be thought as maximal sub-tables with close values. Only few methods address a complete, correct and non-redundant enumeration of such patterns, a well-known intractable problem, while no formal framework exists. We introduce important links between biclustering and Formal Concept Analysis. FCA is known to be, among others, a methodology for biclustering binary data. Handling numerical data is not direct, and we argue that Triadic Concept Analysis (TCA), the extension of FCA to ternary relations, provides a powerful mathematical and algorithmic framework for biclustering numerical data. We discuss hence both theoretical and computational aspects on *biclustering numerical data with triadic concept analysis*. These results also scale to n-dimensional numerical datasets.

We present our major result, which concerns the formalization and efficient mining of biclusters of similar values [88].

```
Mehdi Kaytoue, Sergei O. Kuznetsov, Juraj Macko, Amedeo Napoli:
Biclustering meets triadic concept analysis.
Annals of Mathematics and Artificial Intelligence 70(1-2) - 2014
```

We have also considered biclusters of maximal values on rows or colums [87] and how to formalize and mine them with (i) scaling, (ii) partition pattern structures, and (iii) Triadic Concept Analysis (the results are however not reported in this manuscript):

Mehdi Kaytoue, Víctor Codocedo, Jaume Baixeries, Amedeo Napoli:
Three Interrelated FCA Methods for Mining Biclusters of Similar Values on Columns.
International Conference on Concept Lattices and their Applications (CLA 2014)

2.5 Functional Dependencies

Computing functional dependencies from a relation is an important database topic, with many applications in database management, reverse engineering and query optimization. Whereas it has been deeply investigated in those fields, strong links exist with the mathematical framework of Formal Concept Analysis. Considering the discovery of functional dependencies, it is indeed known that a relation can be expressed as the binary relation of a formal context, whose implications are equivalent to those dependencies. However, this leads to a new data representation that is quadratic in the number of objects w.r.t. the original data. Here, we present an alternative avoiding such a data representation and using pattern structures.

Most importantly, this work allows to consider the discovery of data dependencies as the discovery of pattern implications of a well defined pattern structure. Interestingly, most of the data dependencies presented in a recent survey in the TKDE journal [41], can be expressed in this way. We present our basics, namely *partition pattern structures* and *tolerance pattern structures* and how they formally allow to characterize functional and degenerated multivalued dependencies [20].

Jaume Baixeries, Mehdi Kaytoue, Amedeo Napoli:
Characterizing functional dependencies in FCA with pattern structures.
Annals of Mathematics and Artificial Intelligence 72(1-2) - 2014

The work has then been extended to other kinds of dependencies, namely, approximate matching dependencies [19],

Jaume Baixeries, Mehdi Kaytoue, Amedeo Napoli:
Computing Similarity Dependencies with Pattern Structures.
International Conference on Concept Lattices and their Applications (CLA 2013)

extended to [18],

Jaume Baixeries, Víctor Codocedo, Mehdi Kaytoue, Amedeo Napoli:
Characterizing approximate-matching dependencies in FCA with pattern structures.
Discrete Applied Mathematics 249 - 2018

and order dependencies [46].

Víctor Codocedo, Jaume Baixeries, Mehdi Kaytoue, Amedeo Napoli:
Characterization of Order-like Dependencies with Formal Concept Analysis.
International Conference on Concept Lattices and their Applications (CLA 2016)

2.6 Beyond Pattern Structures

Pattern mining consists in discovering interesting patterns in data. For that, algorithms rely on smart techniques for enumerating the pattern search space and, generally, focus on compressed collections of patterns (e.g. closed patterns), to avoid redundancy. Formal Concept Analysis (FCA) offers a generic framework, called *pattern structures*, to formalize many types of patterns, such as itemsets, intervals, graph and sequence sets. Additionally, it provides generic algorithms to enumerate *all* closed patterns and *only* them. The only condition is that the pattern space is a meet-semilattice, which, unfortunately does not always hold (e.g., for sequential and graph patterns). In this work, we discuss *pattern setups*, a tool that models pattern search spaces relying only on posets. Next, we show that such a framework

Table 1: Toy dataset

ID	a	b	c	$class(.)$
1	150	21	11	l_1
2	128	29	9	l_2
3	136	24	10	l_2
4	152	23	11	l_3
5	151	27	12	l_2
6	142	27	10	l_1

can have some issues since it is too permissive and we propose the new model of *pattern multistructures*; i.e. a model lying between pattern setups and pattern structures that rely on *multilattices*. Subsequently, we revisit some techniques transforming pattern setups to a pattern structure using set of patterns, namely *completions*, and we state *a necessary and sufficient condition* for a pattern setup completion using *antichains* to be a *pattern structure*.

```
Aimene Belfodil, Sergei O. Kuznetsov, Mehdi Kaytoue:
Pattern Setups and Their Completions.
International Conference on Concept Lattices and their Applications (CLA 2018)
```

```
Aimene Belfodil, Sergei O. Kuznetsov, Mehdi Kaytoue:
On Pattern Setups and Pattern Multistructures.
Currently in minor revision to the International Journal of General Systems.
Arxiv: 1906.02963
```

3 Subgroup Discovery and Algorithms

Our main investigation in Formal Concept Analysis had one main goal: properly formalizing data, pattern languages in order theory and showing that many problems (biclustering, pattern mining, subgroup discovery, database dependency discovery) can be actually formalized in a unified framework rooted in Order Theory. We had little concerns on algorithms efficiency and pattern selection (based on, e.g., quality measures). When attempting to make sense of data (see hereafter when applications are presented) through pattern discovery, we got irremediably interested in discriminant pattern discovery. More especially, we considered *Subgroup Discovery*, a leading data mining technique that allows one to elicit descriptions (covering objects called subgroups) that unexpectedly occur, for example, with a class target. Generally, subgroups can be concept extents in FCA, as soon as the unexpectedness can be computed solely given the extent and class frequencies, which represents a vast majority of the existing approaches. Before presenting our contributions, we briefly introduce the problem of subgroup discovery and pattern set discovery.

3.1 A Short Introduction to Subgroup Discovery

For sake of simplicity, we consider here tabular data composed by symbolic and numeric attributes.

Dataset $\mathcal{D}(\mathcal{O}, \mathcal{A}, \mathcal{C}, class)$ Let $\mathcal{O}$, $\mathcal{A}$ and $\mathcal{C}$ be respectively a set of objects, a set of attributes, and a set of class labels. The domain of an attribute $a \in \mathcal{A}$ is $Dom(a)$ where a is either nominal or numerical. The mapping $class : \mathcal{O} \mapsto \mathcal{C}$ associates each object to a unique class label.

Subgroup A subgroup can be represented either by a description or by the set of objects it covers. The description of a subgroup, also called pattern, is given by $d = \langle f_1, \ldots, f_{|\mathcal{A}|} \rangle$ where each f_i is a restriction

on the value domain of the attribute $a_i \in \mathcal{A}$. A restriction for a nominal attribute a_i is a symbol $a_i = v$ with $v \in Dom(a_i)$. A restriction for a numerical attribute a_i is an interval $[l, r]$ with $l, r \in Dom(a_i)$. The description d covers a set of objects called the extent of the subgroup, denoted $ext(d) \subseteq \mathcal{O}$. The support of a subgroup is the cardinality of its extent: $supp(d) = |ext(d)|$.

Subgroup search space The set of all subgroups forms a lattice, denoted as the poset $(\mathcal{S}, \preceq)$. The top is the most general pattern, without restriction. Given any $s_1, s_2 \in \mathcal{S}$, we note $s_1 \prec s_2$ to denote that s_1 is strictly more specific, i.e. it contains more stringent restrictions. If follows that $ext(s_1) \subseteq ext(s_2)$ when $s_1 \preceq s_2$.

The ability of a subgroup to discriminate a class label is evaluated by means of a quality measure. The weighted relative accuracy (WRAcc), intoduced by [102], is among the most popular measures for rule learning and subgroup discovery. Basically, WRAcc considers the precision of the subgroup w.r.t. to a class label relatively to the appearance probability of the label in the whole dataset. This difference is weighted with the support of the subgroup to avoid to consider small ones as interesting.

WRAcc Given a dataset $\mathcal{D}(\mathcal{O}, \mathcal{A}, \mathcal{C}, class)$, the WRAcc of a subgroup d for a label $l \in Dom(\mathcal{C})$ is given by:

$$WRAcc(d, l) = \frac{supp(d)}{|\mathcal{O}|} \times \left(p_d^l - p^l\right)$$

where $p_d^l = \frac{|\{o \in ext(d) | class(o) = l\}|}{supp(d)}$ and $p^l = \frac{|\{o \in \mathcal{O} | class(o) = l\}|}{|\mathcal{O}|}$.

WRAcc returns values in $[-0.25, 0, 25]$, the higher and positive, the better the pattern discriminates the class label. Many quality measures other than WRAcc have been introduced in the literature of rule learning and subgroup discovery (Gini index, entropy, F score, Jaccard coefficient, etc. [3]). Exceptional model mining (EMM) considers multiple labels (label distribution difference in [159], Bayesian model difference in [54], etc.). The choice of a pattern quality measure, denoted φ in what follows, is generally application dependent as explained by [60].

Example 5 *Consider the dataset in Table 1 with objects in $\mathcal{O} = \{1, ..., 6\}$ and attributes in $\mathcal{A} = \{a, b, c\}$. Each object is labeled with a class label from $\mathcal{C} = \{l_1, l_2, l_3\}$. Consider an arbitrary subgroup with description $d = \langle [128 \leq a \leq 151], [23 \leq b \leq 29] \rangle$. Note that, for readability, we omit restrictions satisfied by all objects, e.g., $[9 \leq c \leq 12]$, and thus we denote that $ext(\langle\rangle) = \mathcal{O}$. The extent of d is composed of the objects in $ext(d) = \{2, 3, 5, 6\}$ and we have $WRAcc(d, l_2) = \frac{4}{6}(\frac{3}{4} - \frac{1}{2}) = \frac{1}{6}$. The upper part of the search space (most general subgroups) is given in Figure 4. The direct specializations of a subgroup are given, for each attribute, by adding a restriction: Either by shrinking the interval of values to the left (take the right next value in its domain) or to the right (take the left next value). In this way, the finite set of all intervals taking borders in the attributes domain will be explored (see [90]).*

Subgroup discovery It consists in searching for a set of patterns $\mathcal{R} \subseteq \mathcal{S}$ of high quality on the quality measure φ and whose patterns are not redundant. As similar patterns generally have similar values on φ, we design the pattern set discovery problem as the identification of the local optima w.r.t. φ: Redundant patterns of lower quality on φ are pruned and the extracted local optima are diverse and potentially interesting patterns.

Contributions to Subgroup Discovery

- Facing an application providing multi-labeled data, we proposed a pattern quality measure and enumeration techniques enabling one to discover subgroups that possibly discriminate not only one, but several labels (Guillaume Bosc's PhD).
- Still motivated by an application in neuroscience, we were interested in mining numerical patterns without exhaustive enumeration techniques. Actually, numerical patterns in large dataset (more

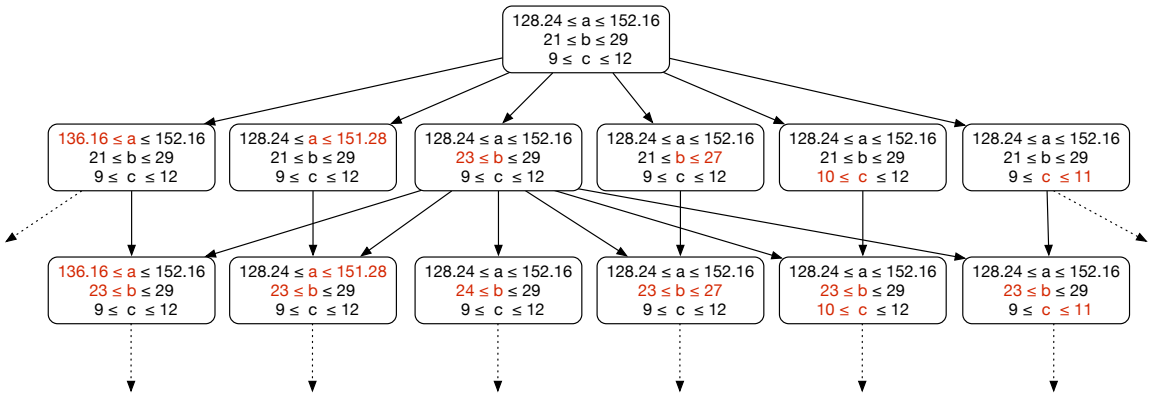


Figure 4: The upper part of the search space for Table 1.

exactly, large search spaces) can be mined with Subgroup Discovery techniques only either with a greedy selection of intervals or an apriori discretization of the data. To mine exact intervals as defined in terms of FCA, we stared to investigate how Monte Carlo Tree Search (MCTS), and its inherent exploration/exploitation trade-off could discover a diverse set of patterns (Guillaume Bosc's PhD).

- MCTS provided excellent results. However, MCTS is an anytime algorithm. It means that we do not expect it to finish (although we prove that it mandatory finishes and visit the whole pattern search space), but we interrupt it at some point. Upon interruption, the user has no idea on the quality of the current result w.r.t to what could be actually expected. We thus proposed a new algorithm paradigm, dedicated to numerical data at present that mines pattern iteratively in finer and finer discretization of the data with several guarantees (Aimene Belfodil's PhD, in collaboration with his brother Adnene).
- Until that moment, our search strategies were aiming at finding good patterns, that is with a high quality measure, and applying post processing to reduce it so that it can be processed by a human expert. MCTS goal is to enhance diversity, that is, sample different part of the search space. However, another approach is to mine pattern sets directly, and optimize the quality of the pattern set directly. We propose such first approach (Aimene Belfodil's PhD, in collaboration with Adnene Belfodil and Anes Bendimerad).
- Our adventure in subgroup discovery was mainly considering categorical and numerical datasets. We started to investigate how MCTS can help subgroup discovery in sequential data (Romain Mathonat's PhD).

3.2 Subgroup Discovery from Multi Labeled Data

In presence of multi-label data, there are many applications for which one is interested in subgroups that differ from the whole dataset only on subsets of labels, and the interesting subsets of labels are not known beforehand: Typically, descriptive rules that conclude on small label sets. SD, and its extension for more complicated target concepts, Exceptional Model Mining (EMM), fail to produce such rules: They consider either the whole set of labels, each label independently, or a unique and fixed label subset chosen *a priori*. We propose to enhance EMM for considering all target subspaces (label subsets): It requires to revisit its formalization, the subgroup search space, and to propose new quality measures expressing how exceptional is a subgroup. Our quality measures consider label distributions dynamically during the pattern exploration that outputs a diversified set of high quality subgroups, whose redundancy is controlled not only on the covered objects, but also on the considered label sets. This is shown through an extensive set of experiments. Finally, we discuss an application in neurosciences which argue the actionability of the discovered subgroups.

Guillaume Bosc, Jérôme Golebiowski, Moustafa Bensafi,
Céline Robardet, Marc Plantevit, Jean-François Boulicaut, Mehdi Kaytoue:
Local Subgroup Discovery for Eliciting and Understanding
New Structure-Odor Relationships.
Discovery Science (DS 2016)

3.3 Mining Large Search Spaces with Best-first Search

The discovery of patterns that accurately discriminate one class label from another remains a challenging data mining task. Subgroup discovery (SD) is one of the frameworks that enables to elicit such interesting patterns from labeled data. A question remains fairly open: How to select an accurate heuristic search technique when exhaustive enumeration of the pattern space is not feasible? Existing approaches make use of beam-search, sampling, and genetic algorithms for discovering a pattern set that is non-redundant and of high quality w.r.t. a pattern quality measure. We argue that such approaches produce pattern sets that lack of diversity: Only few patterns of high quality, and different enough, are discovered. Our main contribution is then to formally define pattern mining as a game and to solve it with Monte Carlo Tree Search (MCTS). It can be seen as an exhaustive search guided by random simulations which can be stopped early (limited budget) by virtue of its *best-first search* property. We show through a comprehensive set of experiments how MCTS enables the anytime discovery of a diverse pattern set of high quality. It outperforms other approaches when dealing with a large pattern search space and for different quality measures. Thanks to its genericity, our MCTS approach can be used for SD but also for many other pattern mining tasks.

Guillaume Bosc, Jean-François Boulicaut, Chedy Raïssi, Mehdi Kaytoue:
Anytime discovery of a diverse set of patterns with Monte Carlo tree search.
Data Min. Knowl. Discov. 32(3): 604-650 - 2018

3.4 Anytime Subgroup Discovery in Numerical Domains with Guarantees

MCTS offers many advantages over other algorithmic paradigms, but sadly does not provide guarantees bounding the *error* of the best pattern quality nor the exploration progression ("*How far are we of an exhaustive search*"). We design here an algorithm for mining numerical data with three key properties w.r.t. the state of the art: (i) It yields progressively interval patterns whose quality improves over time; (ii) It can be interrupted anytime and always gives a guarantee bounding the *error* on the top pattern quality and (iii) It always bounds a distance to the exhaustive exploration.

Aimene Belfodil, Adnene Belfodil, Mehdi Kaytoue:
Anytime Subgroup Discovery in Numerical Domains with Guarantees.
European Conference on Machine Learning and Principles
and Practice of Knowledge Discovery in Databases (ECML/PKDD 2018)
(Best student paper in data mining award).

3.5 Pattern Set Mining

Standard approaches of the literature are based on local pattern discovery, which is known to provide an overwhelmingly large number of redundant patterns. To solve this issue, pattern set mining has been proposed: instead of evaluating the quality of patterns separately, one should consider the quality of a pattern set as a whole. The goal is to provide a small pattern set that is diverse and well-discriminant to the target class. However, most, if not all, pattern set discovery algorithm aim at optimizing the average quality of each pattern making it. We introduce a novel formulation of the task of diverse subgroup set discovery where both discriminant power and diversity of the subgroup set are incorporated in the same quality measure. We propose an efficient and parameter-free algorithm based on a greedy scheme. FSSD

uses several optimization strategies that enable to efficiently provide a high quality pattern set in a short amount of time.

```
Adnene Belfodil, Aimene Belfodil, Anes Bendimerad, Philippe Lamarre,
Céline Robardet, Mehdi Kaytoue and Marc Plantevit.
A Fast and Efficient Algorithm for Subgroup Set Discovery
Data Science and Advanced Analytics (DSAA 2019)
```

3.6 Mining Large Sequences of Itemsets

Though many subgroup discovery algorithms have been proposed for transactional data, discovering subgroups within labeled sequential data and thus searching for descriptions as sequential patterns has been much less studied. In that context, exhaustive exploration strategies can not be used for real-life applications and we have to look for heuristic approaches. We propose the algorithm `SeqScout` to discover interesting subgroups (w.r.t. a chosen quality measure) from labeled sequences of itemsets. This is a new sampling algorithm that mines discriminant sequential patterns using a multi-armed bandit model. It is an anytime algorithm that, for a given budget, finds a collection of local optima in the search space of descriptions and thus subgroups. It requires a light configuration and it is independent from the quality measure used for pattern scoring. Furthermore, it is fairly simple to implement. We provide qualitative and quantitative experiments on several datasets to illustrate its added-value.

```
Romain Mathonat, Diana Nurbakova, Jean-Francois Boulicaut, Mehdi Kaytoue
SeqScout: Using a Bandit Model to Discover Interesting Subgroups in Labeled Sequences.
Data Science and Advanced Analytics (DSAA 2019)
```

4 Knowledge Discovery in Practice

In many applications domains, one uses KDD to try to understand the underlying phenomena and concepts generating some data. I worked mainly on three application domains over the last ten years through long term projects.

Neuroscience Through a long-term collaboration, we proposed a KDD approach with a view to advance the state of the art in understanding the mechanisms of olfaction. We created an interdisciplinary synergy between neuroscientists, chemists and data miners to the emergence of new hypotheses on the links between the structure and the odor of a molecule. Indeed, data-mining methods can be used to answer this discovery problem through descriptive rules discovery in pattern mining [107]. Interestingly, this application problem encountered several data mining challenges that current state-of-the-art-methods were not able to handle (mentioned in the preceding sections).

Social Network Analysis I participated in a three years European project and spent 8 months in an Irish company for that matter. The goal was to apply data mining techniques for real time event detection from social media. I also studied the social network Twitch.tv in 2011 through a characterization social data [93] as I was amazed to see that people like to watch other to play on the Web. This is surprisingly one of my most impactful articles, which can be explained by the fact that Twitch.tv was a few years later bought by Amazon for more than one billion US dollars. It opened also the doors for a one month invited stay followed by a collaboration with the MIT Media Lab [32] and a novel application, video game analytics.

Video Game Analytics I choose to invest some time in this new application domain for several reasons. Games have always been a test bed for AI in general and nowadays massively generate realistic spatiotemporal data which are easily available on the Web. For several industrial projects, data accessibility is a real problem, and video games data can be useful to test our algorithms/methodologies with real goals, needless to add that for assessing a new data mining methodology

or algorithm, one has to apply it on different applications. Finally, attracting students to work in data mining with video game data was also of a non negligible interest.

4.1 Neuroscience & Olfaction

An important goal for understanding the sense of olfaction is to link the perception of smells to the chemistry of odorants. In other words, why do some odorants smell like fruits and others like flowers? While the so-called stimulus-percept issue was resolved in the field of color vision some time ago, the relationship between the chemistry and psycho-biology of odors remains unclear up to the present day. Although a series of investigations have demonstrated that this relationship exists, the descriptive and explicative aspects of the proposed models that are currently in use require greater sophistication. One reason for this is that the algorithms of current models do not consistently consider the possibility that multiple chemical rules can describe a single quality despite the fact that this is the case in reality, whereby two very different molecules can evoke a similar odor. Moreover, the available datasets are often large and heterogeneous, thus rendering the generation of multiple rules without any use of a computational approach overly complex. We considered these two issues. First, we built a new database containing 1689 odorants characterized by physicochemical properties and olfactory qualities. Second, we developed a computational method based on a subgroup discovery algorithm that discriminated perceptual qualities of smells on the basis of physicochemical properties. Third, we ran a series of experiments on 74 distinct olfactory qualities and showed that the generation and validation of rules linking chemistry to odor perception was possible. Taken together, our findings provide significant new insights into the relationship between stimulus and percept in olfaction. In addition, by automatically extracting new knowledge linking chemistry of odorants and psychology of smells, our results provide a new computational framework of analysis enabling scientists in the field to test original hypotheses using descriptive or predictive modeling.

```
Carmen C. Licon, Guillaume Bosc, Mohammed Sabri, Marylou Mantel,
Arnaud Fournel, Caroline Bushdid, Jerome Golebiowski, Celine Robardet,
Marc Plantevit, Mehdi Kaytoue, Moustafa Bensafi
Chemical features mining provides new descriptive structure-odor relationships.
PLoS Computational Biology 15(4) - 2019
```

4.2 Video Game Analytics

"Electronic-sport" (E-Sport) is now established as a new entertainment genre. More and more players enjoy streaming their games, which attract even more viewers. In fact, social studies report that casual players were found to prefer watching professional gamers rather than playing the game themselves. In this context, advertising provides a significant source of revenue to the professional players, the casters (displaying other people's games) and the game streaming platforms. For this paper, we crawled, during more than 100 days, the most popular among such specialized platforms: Twitch.tv. Thanks to these gigabytes of data, we propose a first characterization of a new Web community, and we show, among other results, that the number of viewers of a streaming session evolves in a predictable way, that audience peaks of a game are explainable and that a Condorcet method can be used to sensibly rank the streamers by popularity. Last but not least, we hope that this paper will bring to light the study of E-Sport and its growing community. They indeed deserve the attention of industrial partners (for the large amount of money involved) and researchers (for interesting problems in social network dynamics, personalized recommendation, sentiment analysis, etc.).

```
Mehdi Kaytoue, Arlei Silva, Loïc Cerf, Chedy Raïssi, Wagner Meira Jr.:
Watch me playing, i am a professional: a first study on video game live streaming.
Int. Conf. World Wide Web (Companion Volume: Workshops 2012)
```

Noticing and trying to understand this phenomena at its early stage led us to the following thoughts that the video game industry has to deal with. To be successful (that is, generate revenue), a game has

to consider the very delicate trade-off between difficulty and entertainment. Indeed, from a player point of view, a game is attractive for the pro-players if it is difficult enough, and not too easy for the casual player. The rules have to be simple and easy to understand. From a spectator point of view, the game should allow the professional players to be able to show their skills, that is, highlight strategies and game plays that the casual player is not able to perform. In the end, the game has to be designed to satisfy both casual and pro players: the casual player needs to feel to learn and improve, the pro can show capacities the amaze the others. Social media such as Twitch.tv, and their audience, allow to measure precisely the interest of the casual players, advertise... This clearly explains why it was bought by Amazon: a new, simple and precise tool to help recommend and sell on Amazon video game related products.

We decided to investigate several key points involved in the above mentioned trade-off. One of the main reason was that several industrial projects we had were actually (and surprisingly!) lacking of data. Game data became at the moment more and more freely available on the Web and traducing spatio-temporal data with a lot of meta-data, most of them created by different online gamer communities. Luckily, the different problems we had for these projects could translate in video game settings.

The different aspects we got interested in are the three following. Notice that the first one was mainly dealt with during my stay at the MIT Media Lab and as a playground in the early and pre stages of the PhD thesis of Guillaume Bosc (before that he focused on olfactory issues, as detailed previously). The second was achieved in collaboration during the PhD of Olivier Cavandenti as an application. The third part was achieved during the PhD of Olivier Cavadenti and the post doctoral studies of Víctor Codocedo.

Characterizing player strategies Starting from large game logs of data, containing only the actions made by the players, we show how sequential pattern mining can extract interesting strategies from real time strategy games (RTS) such as StarCraft II. Such patterns can be used to (i) study if the rules of the game are balanced and adapt them otherwise, (ii) study the strategies of a players (e.g., before confronting him in a tournament), and (iii) study its own strategies to learn and improve. We believe that our approach can become a basic tool for *balance designers* when analyzing a subset of historical data of a game in beta phase, or even after its release, through an exploratory process (KDD and interactive mining); but also for electronic sports coaches for analyzing and modeling opponents. Finally, whereas Google Deep Mind very recently proposed an automated agent (Alpha Star) that was in some ways able to win against a Human professional player of StarCraft II [13], discovering patterns that the AI would play (e.g. after generating thousands of AI vs. AI games), could lead to the discovery of interesting unknown strategies.

```
Guillaume Bosc, Mehdi Kaytoue, Chedy Raïssi,
Jean-François Boulicaut, Philip Tan:
Mining Balanced Sequential Patterns in RTS Games.
European Conference on Artificial Intelligence (ECAI 2014)
```

extended as a journal version as

```
Guillaume Bosc, Philip Tan, Jean-François Boulicaut, Chedy Raïssi, Mehdi Kaytoue:
A Pattern Mining Approach to Study Strategy Balance in RTS Games.
IEEE Trans. Comput. Intellig. and AI in Games 9(2) - 2017
```

Actually, our first attempt to extract knowledge from StarCraft II game data was the following

```
Cécile Low-Kam, Chedy Raïssi, Mehdi Kaytoue, Jian Pei:
Mining Statistically Significant Sequential Patterns.
International Conference on Data Mining (ICDM 2013)
```

Helping casual players to improve The success of electronic sports (eSports), where professional gamers participate in competitive leagues and tournaments, brings new challenges for the video game industry. Other than fun, games must be difficult and challenging for eSports professionals

but still easy and enjoyable for amateurs. In this work, we consider *Multi-player Online Battle Arena* games (MOBA) and particularly, "Defense of the Ancients 2", commonly known simply as DOTA2. In this context, a challenge is to propose data analysis methods and metrics that help players to improve their skills. We design a data mining-based method that discovers strategic patterns from historical behavioral traces: Given a model encoding an expected way of playing (the norm), we are interested in patterns deviating from the norm that may explain a game outcome from which player can learn more efficient ways of playing. The method is formally introduced and shown to be adaptable to different scenarios. Finally, we provide an experimental evaluation over a dataset of $10,000$ behavioral game traces.

```
Olivier Cavadenti, Víctor Codocedo, Jean-François Boulicaut, Mehdi Kaytoue:
What Did I Do Wrong in My MOBA Game?
Mining Patterns Discriminating Deviant Behaviours.
IEEE Int. Conf. on Data Science and Advanced Analytics (DSAA) - 2016
```

Identifying anonymous players on the Web In e-sports, cyberathletes conceal their online training using different aliases or avatars (virtual identities), which allow them not being recognized by the opponents they may face in future competitions (with cash prices challenging already most of the traditional sports). We show that behavioral data generated by the games allows predicting the avatar associated to a game play with high accuracy. However, when a player uses several avatars, accuracy drastically drops as prediction models cannot easily differentiate the player's different avatar aliases. Since mappings between players and avatars do not exist, we introduce the *avatar aliases identification problem* and propose an original approach for alias resolution based on supervised classification and Formal Concept Analysis. We thoroughly evaluate our method with the video game StarCraft II which has a very wide and active community with players from diverse cultures and nations. We show that under some circumstances, the avatars of a given player can easily be recognized as such. These results are valuable for e-sport structures (to help preparing tournaments), and game editors (detecting cheaters or usurpers).

```
Olivier Cavadenti, Víctor Codocedo, Jean-François Boulicaut, Mehdi Kaytoue:
When cyberathletes conceal their game:
Clustering confusion matrices to identify avatar aliases.
IEEE Int. Conf on Data Science and Advanced Analytics (DSAA) - 2015
```

4.3 Social Networks

Social networks (such as Twitter, Instagram, ...) are rich sources of information that can be used to build a huge number of applications and services for end-users (b2c), for companies, e.g. with analytics platforms (b2b), but also to help governments and charitable organizations. Through several public APIs, one can access streams of messages, often provided with text (including hashtags, user mentions and URIs), media (images or video) and geo-tags indicating the position of the user emitting the message (called *post* in the sequel).

One way of exploiting such data is to discover global trends and detecting events in the streams of posts. The motivations are manifold: disaster detection, epidemic surveillance, identification of news-worthy events that traditional media are slow to pick up, identification of trends, monitoring of brand perception, etc. The question of how to identify events in streams of text data has been a research topic for more than a decade now, starting from e-mail, via blog posts, to location-based social networks data [150]. The general idea underlying most of that work is identifying "bursty" topics (mentioned significantly more often during a time period than in the period preceding it).

Whereas most of the existing systems identify global trends, only a few take into account the geo-localization of the posts for detecting local events [168]. This is actually the goal of Gazouille: harvesting data from urban areas, the system is able to detect spatially circumscribed events in real-time and

to intelligibly characterize them (with their periodicity, users, key-words, and via a media gallery, e.g. in Figure 5).

```
Pierre Houdyer, Albrecht Zimmermann, Mehdi Kaytoue,
Marc Plantevit, Joseph Mitchell, Céline Robardet:
Gazouille: Detecting and Illustrating Local Events from Geolocalized
Social Media Streams.
European Conference on Machine Learning and Principles and
Practice of Knowledge Discovery in Databases (ECML/PKDD 2015) - Demo paper
```

The Gazouille plateform is actually one of the results of a three years European project[3]. I spent 8 months in an Irish company for that matter. This project, which main goal was *live event detection and characterization*, was the occasion to collaborate within the research group I had just joined in the LIRIS laboratory, on themes that were new for me, such as *graph mining*. We can note two main results from this collaboration.

Triggering patterns of topology changes in dynamic graphs To describe the dynamics taking place in networks that structurally change over time, we propose an approach to search for vertex attributes whose value changes impact the topology of the graph. In several applications, it appears that the variations of a group of attributes are often followed by some structural changes in the graph that one may assume they generate. We formalize the triggering pattern discovery problem as a method jointly rooted in sequence mining and graph analysis. We apply our approach on three real-world dynamic graphs of different natures – a co-authoring network, an airline network, and a social bookmarking system – assessing the relevancy of the triggering pattern mining approach.

```
Mehdi Kaytoue, Yoann Pitarch, Marc Plantevit, Céline Robardet:
Triggering patterns of topology changes in dynamic graphs.
Int. Conf. on Advances in Social Network Analysis and Mining (ASONAM 2014)
```

extended to a journal version

```
Mehdi Kaytoue, Yoann Pitarch, Marc Plantevit, Céline Robardet:
What effects topological changes in dynamic graphs?
Elucidating relationships between vertex attributes and the graph structure.
Social Netw. Analys. Mining 5(1) - 2015
```

Exceptional Subgraph Mining Many relational data result from the aggregation of several individual behaviors described by some characteristics. For instance, a bike-sharing system may be modeled as a graph where vertices stand for bike-share stations and connections represent bike trips made by users from one station to another. Stations and trips are described by additional information such as the description of the geographical environment of the stations (business vs. residential area, closeness to POI, elevation, urbanization density, etc.), or properties of the bike trips (timestamp, user profile, weather, events and other special conditions about the trip). Identifying highly connected components (such as communities or quasi-cliques) in this graph provides interesting insights into global usages but does not capture mobility profiles that characterize a subpopulation. To tackle this problem we proposed an approach rooted in exceptional model mining to find exceptional contextual subgraphs, i.e., subgraphs generated from a context or a description of the individual behaviors that is exceptional (behaves in a different way) compared to the whole augmented graph. We experimented this approach with different kinds of datasets, such as social network data and video game log data.

```
Mehdi Kaytoue, Marc Plantevit, Albrecht Zimmermann, Ahmed Anes Bendimerad,
Céline Robardet: Exceptional contextual subgraph mining.
Machine Learning 106(8) - 2017
```

[3]GRAISearch: `https://cordis.europa.eu/project/rcn/192400/factsheet`

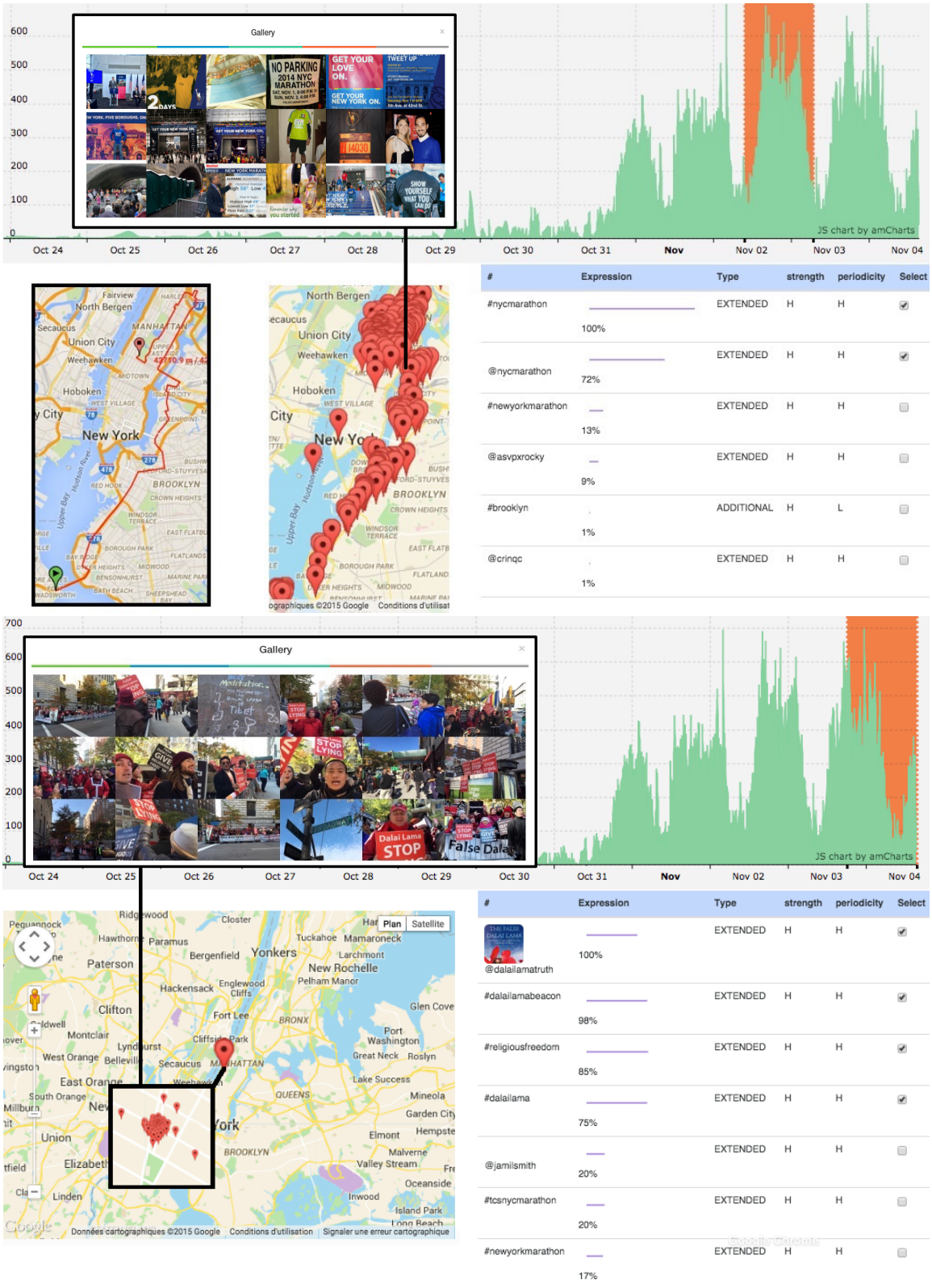


Figure 5: Detecting events in New-York: the NYC 2014 marathon (top), a protest (bottom)

5 Outline

The manuscript does not cover all the contributions mentioned beforehand. Instead, it is structured into three main parts covering each one of my main axis of research and each part is based on a selection of two to three articles. Each chapter has been thus adapted and shortened from the original publication, so that the reader can have a self contained view of each of these works, and understand the challenges at the time of the realization of the work: we cover here a period spanning from 2011 to 2019.

Data and Pattern Formalization with FCA The first part presents three examples of pattern characterization with FCA, namely *biclusters*, *database dependencies* and *numerical patterns*. The message reflected by the selection is the following: *FCA can formally model many types of patterns and there is still room for research.*

Pattern Mining Algorithms We focus in this part on *anytime subgroup discovery algorithms*, that is, algorithms that can be interrupted at any moment to produce a result to the expert. Moreover, the more the budget/time allowed, the better the result. The choice of the two articles concern our different attempts to define or adapt algorithmic paradigms for pattern mining (subgroup discovery in particular). First, we explain how *Monte Carlo Tree Search* can be used to successfully discover a diverse set of patterns. Second, we consider numerical data in particular and propose an algorithm that exploits the fact that interval patterns can be found more and more precisely in finer and finer discretization of the data.

KDD Methodologies and Applications Finally, we focus on one application only: *Video Game Analytics*. We present two use cases where KDD and pattern mining help to discover valuable knowledge from game data. Firstly, we explain how sequential patterns from action game data reveal player strategies and game misconception. Then, after showing that a player can be predicted from his keyboard usage (as for typing free and fixed texts with keystrokes analytics), we detail how FCA can discover the fact that a player uses different names (that is, two labels in predictive settings actually correspond to the same individual).

Part I

Formal Concept Analysis

Chapter 1

Biclustering

In this chapter, we introduce first the task of Biclustering (Section 1.1), before focusing on one type of biclusters: biclusters of similar values (Section 1.2). Given a numerical data table and a similarity parameter, such biclusters are defined as maximal rectangles of pairwise similar values (modulo rows/columns permutations). We formalize these patterns within Formal Concept Analysis (Section 1.4) and present how to compute them efficiently (Section 1.5) before concluding with perspectives (Section 1.6). Note that our experimental results are not be discussed here but are available [88].

1.1 Introduction

Taking roots in the work of Hartigan [76] in 1972 and extended by Mirkin in 1996 [117], numerical data biclustering then strongly attracted attention from the beginning of 2000's as a first answer to new challenges raised by gene expression data analysis [44] and recommender systems design [4]. Starting from an object/attribute numerical data-table, the goal is to group together some objects with some attributes according to the values taken by these attributes for these objects. The main idea of biclustering is to overcome the limitation of standard clustering techniques producing partitions of objects where distance functions that use all the attributes may be ineffective and hard to interpret [7]. For example, in gene expression data, it is known that genes (objects) may share a common behavior for a subset of biological situations (attributes) only: one should accordingly produce local patterns to characterize biological processes, the latter should possibly overlap, since a gene may be involved in several processes. The same remark applies for recommender systems, where the taste of users for some items is realized by a so-called utility matrix (usually very sparse): one is interested in local patterns characterizing groups of users that strongly share almost the same tastes for a subset of items [4].

Accordingly, a bicluster is formally defined as a pair composed of a set of objects and a set of attributes. Such a pair can be represented as a rectangle in a numerical table, modulo rows and columns permutations. Table 1.1 is a numerical dataset with objects in rows and attributes in columns, while each table entry corresponds to the value taken by the attribute in column for the object in row. Table 1.2 illustrates bicluster $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ as a grey rectangle that can be understood as a sub-table of the original one. There are several types of biclusters in the literature, depending on the relation between the values taken by their attributes for their objects (as surveyed by Madeira and Oliveira [111]). The most simple case can be understood as rectangles of equal values: a bicluster corresponds to a set of objects whose attributes take exactly the same value, e.g. $(\{g_1, g_2, g_3\}, \{m_5\})$. Constant biclusters only appear in idyllic situations. Accordingly, a straightforward generalization of such biclusters lies in so-called biclusters of similar values: they are represented by rectangles with almost identical, say similar, values (see [111, 25, 89] and to a similar extent [39]). Table 1.2 illustrates a bicluster of similar values $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ where two values are said similar if their difference is lower than 1. Moreover, this bicluster is maximal: neither an object nor an attribute can be added without violating the similarity

Table 1.1: 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 1.2: 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

condition. The problem of biclustering that we investigate in this paper consists in extracting all pairs (A, B), such that A and B are maximal sets with respect to a similarity constraint between values.

To better understand our investigation, we recall a definition of bicluster of Prelic et al. in binary data or relation, i.e. an object has or not an attribute [137]: *inclusion-maximal biclusters* are defined as maximal sets of objects related to a maximal set of attributes. As shown in [94], this definition exactly meets the one of *formal concepts* in the Formal Concept Analysis theory (FCA, [64]). Hence, our general intuition is that FCA can be used to answer the problem of biclustering numerical data, which is not straightforward, FCA basically applying to binary data.

Formal Concept Analysis is a branch of applied lattice theory that appeared in the 1980's [165, 64] and proved to be very useful in data analysis. It aims at representing data as a formal concept hierarchy, the later being useful for many tasks of, among others, knowledge management and data-mining [166, 156, 138, 11]. Starting from a binary relation between a set of objects and their attributes, so-called formal concepts are built as maximal sets of objects in relation with a maximal set of attributes. If we represent the binary relation as a binary table (with objects as rows, attributes as columns and 0/1 as values if an object has/has not an attribute), a formal concept is represented as a maximal rectangle of 1 values. The ordering of concepts among a complete lattice makes overlapping of such local and maximal patterns natural. Then a complete enumeration of patterns respecting some constraints like closure and minimal frequency is possible [27, 99]. Indeed, the subsets of patterns satisfying these constraints is an order ideal of the lattice of patterns.

It is now natural to argue that FCA can be considered as a kind of biclustering method for binary data. As such, it has been applied to numerical data, and especially to gene expression data after an adequate transformation, see e.g. [134, 137, 27, 125]. The process that turns numerical data into binary data (discretization), usually called conceptual scaling in FCA, generally comes with a loss of information, and thus the obtained formal concepts are not exactly and formally related with biclusters (although they are good representatives). This being stated, biclustering binary data is still attracting a lot of attention, to cope with several issues such as the number of produced patterns and enabling a fault tolerance to leverage the strict notion of maximality of formal concepts, see e.g. [118, 24, 43, 119, 82]. Biclustering directly numerical data, without a priori binarization, has also been widely studied, and several ad hoc algorithms have been proposed to extract specific kind of biclusters with different algorithmic strategies (such as *divide-and-conquer*, *greedy iterative search*, *exhaustive enumeration* as deeply surveyed in [111]). Indeed enumerating all biclusters of a given type is an intractable problem and complete approaches generally fail. Our main contribution states that such approach is possible when considering the problem of extracting maximal bicluster of similar values in formal concept analysis settings, outperforming the other existing algorithms for this task [25, 89]. Other concerns of biclustering are to be able to consider multi-dimensional data (e.g. when the expression of a gene is monitored in several situations across time [173, 153]) and parallelization of the algorithms [34] which both are important issues we address in this paper. This leads us to our main contributions.

Problem. We consider here maximal biclusters of similar values, denoted by (A, B) where A and B are respectively maximal sets of objects and attributes, such that the values taken by these attributes for these objects are pairwise similar. Given a similarity parameter θ, the similarity relation is defined as $a \simeq_\theta b \iff |b - a| \leq \theta$, for any numbers a and b. The problem is to design an approach that allows an exact, correct and complete extraction of maximal biclusters of similar values.

Contribution 1. Triadic Concept Analysis (TCA) [104] is an extension of FCA to handle ternary

relations: an object has an attribute under a given condition. This leads to triadic contexts, i.e. data are represented as a "box", where so-called triadic concepts can be seen as maximal sets of objects in relation with a maximal set of attributes under a maximal set of conditions, i.e. a maximal "sub-box" of $\times$ in the context (still with rows, columns and layers permutations). We show then, that after turning the original numerical data in a triadic context without loss of information (with interordinal scaling [64]), the resulting triadic concepts are in 1-1-correspondence with the maximal biclusters of similar values for any similarity parameter θ (stating if two values are similar or not). Then, such concepts can be organized in a trilattice whose diagram gives a visualization of biclusters in the numerical dataset. Finally, we show that this result naturally holds when considering n-dimensional numerical datasets.

Contribution 2. Maximal biclusters of similar values for a user-defined similarity parameter have been studied with complete approaches in [25, 89]. In [25], an algorithm for extracting such biclusters is presented, while [89] shows how such biclusters can be characterized by post-processing a concept lattice built from the numerical data directly. We show that our first contribution can be easily adapted to answer this problem, with a new generic algorithm TriMax that shows better results than its competitors and can be naturally parallelized.

In other words, firsly, theoretical new links are emphasized between biclustering and FCA in general, and TCA in particular, for a better understanding of numerical pattern mining with closure operators. Secondly, a computational aspect is investigated using these links: it allows one to bring back a problem of biclustering into well known-settings (i.e. FCA and pattern-mining) and comes with better computational properties and several perspectives of research.

1.2 Problem Settings

A numerical dataset is formalized by a many-valued context [64] and we define accordingly (maximal) biclusters of similar values.

Definition 1 (Many-valued context) *(G, M, W, I) is called many-valued context, or simply numerical dataset in this paper, with G being a set of objects, M a set of attributes, W the set of attribute values and I a ternary relation defined on $G \times M \times W$. The fact $(g, m, w) \in I$, also written $m(g) = w$, means that "Attribute m takes the value w for the object g".*

Example 6 *Table 1.1 is a numerical dataset, or many-valued context, with objects $G = \{g_1, g_2, g_3, g_4\}$, attributes $M = \{m_1, m_2, m_3, m_4, m_5\}$, attribute values $W = \{0, 1, 2, 6, 7, 8, 9\}$ and $m_5(g_2) = 6$.*

Definition 2 (Bicluster) *Given (G, M, W, I), a bicluster is a tuple (A, B) with $A \subseteq G$ and $B \subseteq M$.*

Definition 3 (Similarity relation and bicluster of similar values) *Let $w_1, w_2 \in W$ be two attribute values and $\theta \in \mathbb{R}$ be a user-defined parameter, called similarity parameter or threshold. w_1 and w_2 are said to be similar iff $|w_1 - w_2| \leq \theta$, which we denote by $w_1 \simeq_\theta w_2$. (A, B) is bicluster of similar values if $m(g) \simeq_\theta n(h)$ for all $g, h \in A$ and for all $m, n \in B$.*

Definition 4 (Maximal bicluster of similar values) *A bicluster of similar values (A, B) is maximal if adding either an object in A or an attribute in B does not result in a bicluster of similar values.*

Example 7 (From Table 1.1) *$(\{g_1, g_4\}, \{m_2, m_4\})$ is a bicluster. $(\{g_1, g_2\}, \{m_2\})$ is a bicluster of similar values with $\theta \geq 1$. However, it is not maximal. With $1 \leq \theta < 5$, $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ is maximal. Finally, with $\theta = 7$ the bicluster $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3, m_4, m_5\})$ is maximal. Note that a constant (maximal) bicluster is a (maximal) bicluster of similar values with $\theta = 0$.*

Thus the problem that we address in this article is the extraction of all maximal biclusters of similar values from a numerical dataset. We desire the extraction to be complete, correct and non-redundant compared to most of existing methods of the literature based on heuristics [111]. We will show that FCA is a good candidate as a formal framework for such a task.

Table 1.3: A triadic context (G, M, B, Y) with the triadic concept $(\{g_3, g_4\}, \{m_2, m_3\}, \{b_1, b_2, b_3\})$

b_1

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

b_2

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

b_3

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

1.3 Triadic Concept Analysis

Lehmann and Wille introduced Triadic Concept Analysis (TCA [104]) to handle ternary relations between some objects, attributes and conditions. Data are formalized by a triadic context from which triadic concepts are extracted and organized as a trilattice.

Definition 5 (Triadic context) *Data are represented by a triadic context $\mathbb{K} = (G, M, B, Y)$, where G, M, and B are respectively called sets of objects, attributes and conditions, and $Y \subseteq G \times M \times B$. The fact $(g, m, b) \in Y$ is interpreted as the statement "Object g has attribute m under condition b".*

Example. An example of such triadic context lies in Table 1.3 where the very first cross (to the left) denotes the fact "Object g_2 has attribute m_1 under the condition b_1, i.e. $(g_2, m_1, b_1) \in Y$. In this tabular representation, each table corresponds to the projection of the triadic context for one condition. Another choice could have been made.

Definition 6 (Triadic concept) *A triadic concept of (G, M, B, Y) is a triple (A_1, A_2, A_3) with $A_1 \subseteq G$, $A_2 \subseteq M$ and $A_3 \subseteq B$ satisfying the two following statements: (i) $A_1 \times A_2 \times A_3 \subseteq Y$, $X_1 \times X_2 \times X_3 \subseteq Y$ and (ii) $A_1 \subseteq X_1$, $A_2 \subseteq X_2$ and $A_3 \subseteq X_3$ implies $A_1 = X_1$, $A_2 = X_2$ and $A_3 = X_3$. If (G, M, B, Y) is represented by a three dimensional table, (i) means that a concept stands for a 3-dimensional rectangle full of crosses while (ii) characterizes component-wise maximality of concepts. For a triadic concept (A_1, A_2, A_3), A_1 is called the extent, A_2 the intent and A_3 the modus.*

Example. $(\{g_3, g_4\}, \{m_2, m_3\}, \{b_1, b_2, b_3\})$ is a triadic concept in the triadic context represented by Table 1.3. Representing the triadic context as a cube, where each condition is a slice, one can observe that this triadic concept denotes a maximal cube of crosses (modulo lines, columns and slices permutations).

Definition 7 (Outer derivation operators) *To describe the derivation operators, it is convenient to represent a triadic context as (K_1, K_2, K_3, Y). Then, for $\{i, j, k\} = \{1, 2, 3\}$, $j < k$, $X \subseteq K_i$ and $Z \subseteq K_j \times K_k$, (i)-derivation operators are defined by:*

$\Phi : X \to X^{(i)}$ *with for all* $a_i \in X$, $\Phi(a_i) = \{(a_j, a_k) \in K_i \times K_j \mid (a_i, a_j, a_k) \in Y\}$

$\Phi' : Z \to Z^{(i)}$ *with for all* $(a_j, a_k) \in Z$, $\Phi'(a_j, a_k) = \{a_i \in K_i \mid (a_i, a_j, a_k) \in Y\}$

This definition leads to dyadic contexts

$$\begin{aligned} \mathbb{K}^{(1)} &= \langle K_1, K_2 \times K_3, Y^{(1)} \rangle \\ \mathbb{K}^{(2)} &= \langle K_2, K_1 \times K_3, Y^{(2)} \rangle \\ \mathbb{K}^{(3)} &= \langle K_3, K_1 \times K_2, Y^{(3)} \rangle \end{aligned}$$

where

$$gY^1(m, b) \iff mY^2(g, b) \iff bY^3(g, m)$$

Example. Consider $i = 1$, $j = 2$ and $k = 3$, i.e. $K_1 = G$, $K_2 = M$ and $K_3 = B$. Given an arbitratry set of objects $X = \{g_4\}$, we have:

$$\begin{aligned} \Phi(X) &= \{(m_2, b_1), (m_3, b_1), (m_2, b_2), (m_3, b_2), (m_2, b_3), (m_3, b_3)\} \\ \Phi'\Phi(X) &= \{g_3, g_4\} \end{aligned}$$

Definition 8 (Inner derivation operators) *Further derivation operators are defined as follows: for* $\{i,j,k\} = \{1,2,3\}$*,* $X_i \subseteq K_i$*,* $X_j \subseteq K_j$ *and* $A_k \subseteq K_k$*, the* (i,j,A_k)*-derivation operators are defined by:*

$\Psi : X_i \to X_i^{(i,j,A_k)}$ *with for all* $(a_i,a_k) \in X_i \times A_k$*,* $\Psi(a_i,a_k) = \{a_j \in K_j \mid (a_i,a_j,a_k) \in Y\}$

$\Psi' : X_j \to X_j^{(i,j,A_k)}$ *with for all* $(a_j,a_k) \in X_j \times A_k$*,* $\Psi'(a_j,a_k) = \{a_i \in K_i \mid (a_i,a_j,a_k) \in Y\}$

This definition yields the derivation operators of dyadic contexts defined by

$$\begin{array}{rcl} \mathbb{K}_{A_k}^{ij} & = & \langle K_i, K_j, Y_{A_k}^{ij} \rangle \\ \text{where } (a_i,a_j) \in Y_{A_k}^{ij} & \iff & a_i, a_j, a_k \text{ are related by } Y \text{ for all } a_k \in A_k \end{array}$$

Example. Consider $i = 1$, $j = 2$ and $k = 3$, i.e. $K_1 = G$, $K_2 = M$ and $K_3 = B$, $A_3 = \{b_1, b_2\}$ and $X = \{g_3\}$:

$$\Psi(X) = \{m_2, m_3\} \qquad\qquad \Psi'\Psi(X) = \{g_3, g_4\}$$

Operators Φ and Φ' will be called outer operators, pair of both operators outer closure and dyadic operators Ψ and Ψ' inner operators or inner closure when pair of both is used.

Definition 9 (Triadic concept formation) *A concept having* X_1 *in its extent can be constructed as follows.*

$$(X_1^{(1,2,A_3)(1,2,A_3)}, X_1^{(1,2,A_3)}, (X_1^{(1,2,A_3)} \times X_1^{(1,2,A_3)})^{(3)}))$$

Example. In the previous example, we have $(\{g_3, g_4\}, \{m_2, m_3\}, \{b_1, b_2, b_3\})$.

From a computational point of view, [83] developed the algorithm TRIAS for extracting frequent triadic concepts, i.e. whose extent, intent and modus cardinalities are higher than user-defined thresholds (see also [84]). Cerf et al. presented a more efficient algorithm called DATA-PEELER able to handle n-ary relations [43] while formal definitions lie in so-called Polyadic Concept Analysis [162].

1.4 Biclusters of Similar Values in Triadic Concept Analysis

This first contribution considers the problem of generating maximal biclusters for any θ with TCA after a scaling procedure. We then show how to represent the resulting set of concepts with line diagrams, and extend the methodology to n-dimensional numerical datasets.

1.4.1 Scaling numerical data into a triadic context

Starting from a numerical dataset (G, M, W, I), the basic idea lies in building a triadic context (G, M, T, Y) where the two first dimensions remain formal objects and formal attributes, while W is scaled into a third dimension denoted by T. This new dimension T is called the *scale dimension*: intuitively, it gives different "spaces of values" that each object-attribute pair $(g, m) \in G \times M$ can take. Once the scale is given, a triadic context is derived and it gives rise to triadic concepts.

We use the *interordinal scaling* [64] to build the scale dimension. It allows one to encode in 2^T all possible intervals of values in W. This scale allows one to derive a triadic context from which any bicluster of similar values can be characterized as a triadic concept. We make these statements more precise and illustrate the whole procedure with examples.

Definition 10 (Interordinal Scaling) *A scale is a binary relation* $J \subseteq W \times T$ *associating original elements from the set of values* W *to their derived elements in* T*. In the case of interordinal scaling,* $T = \{[min(W), w], \forall w \in W\} \cup \{[w, max(W)], \forall w \in W\}$*. Then* $(w,t) \in J$ *iff* $w \in t$*.*

J	$t_1 = [0,0]$	$t_2 = [0,1]$	$t_3 = [0,2]$	$t_4 = [0,6]$	$t_5 = [0,7]$	$t_6 = [0,8]$	$t_7 = [0,9]$	$t_8 = [1,9]$	$t_9 = [2,9]$	$t_{10} = [6,9]$	$t_{11} = [7,9]$	$t_{12} = [8,9]$	$t_{13} = [9,9]$
0	×	×	×	×	×	×	×						
1		×	×	×	×	×	×	×					
2			×	×	×	×	×	×	×				
6				×	×	×	×	×	×	×			
7					×	×	×	×	×	×	×		
8						×	×	×	×	×	×	×	
9							×	×	×	×	×	×	×

Table 1.4: Interordinal scale of the set of attribute values W.

Example 8 *Table 1.4 gives the tabular representation of the interordinal scale for Table 1.1. Each row describes a single value, while dyadic concepts represent all possible intervals over W. An example of dyadic concept in this table is given by $(\{6,7,8\}, \{t_6, t_7, t_8, t_9, t_{10}\})$, rewritten as $(\{6,7,8\}, \{[6,8]\})$ since $\{t_6, t_7, t_8, t_9, t_{10}\}$ represents the interval $[0,8] \cap [0,9] \cap [1,9] \cap [2,9] \cap [6,9] = [6,8]$.*

Definition 11 (Triadic scaled context) *Let Y be a ternary relation $Y \subseteq G \times M \times T$. Then $(g,m,t) \in Y$ iff $(m(g),t) \in J$, or simply $m(g) \in t$. We call the tuple (G,M,T,Y) the triadic scaled context of the numerical dataset (G,M,W,I).*

Example 9 *The object-attribute pair (g_1, m_1) taking value $m_1(g_1) = 1$ is scaled into triples $(g_1, m_1, t) \in Y$, where t takes any interval in $\{[0,1], [0,2], [0,6], [0,7], [0,8], [0,9], [1,9]\}$. The intersection of intervals in this set is the original value itself, i.e. $m_1(g_1) = 1$, a basic property of interordinal scaling. As a result, Table 1.5 illustrates the whole scaled triadic context derived from the numerical dataset given in Table 1.1 using interordinal scaling. The very first cross (×) in this table (upper left) represents the tuple (g_2, m_4, t_1), meaning that $m_4(g_2) \in [0,0]$.*

We present now our first main result: there is a one-to-one correspondence between (i) the set of maximal biclusters of similar values in a given numerical dataset for any similarity parameter θ and (ii) the set of all triadic concepts in the triadic context derived with interordinal scaling. Consider first the following definition and notations.

Definition 12 (Standard order of interordinal scale attributes) *The values of the interordinal scale are intervals. Define the* standard order *on $2k-1$ attributes of the interordinal scale based on k first natural numbers as follows: $[1,1], [1,2], \ldots, [1,k], [2,k], \ldots, [k,k]$. Having the standard order on the attributes of the interordinal scale one can think of attributes having numbers from 1 to $2k+1$. Note the obvious* main property of the standard order on attributes of the interordinal scale*: if an object has two scale attributes with numbers r and s, $r < s$, then it has all scale attributes with numbers in $[r,s]$.*

For a many-valued context (G,M,W,I), let the set W ($|W| = q$) be the set of numerical values enumerated in the ascending order from 1 to q, and let $g(m)$ be a map taking attribute m to its value $w \in W$ for object g. Let the numerical values from W be interordinally scaled with the standard order on the scale attributes, so we can denote the scale attributes by $m_1, \ldots, m_q, \ldots, m_{2q-1}$. Let $B = \{m_1, \ldots, m_q, \ldots m_{2q-1}\}$ and (G,M,B,Y) be the triadic context such that $(g,m,b) \in Y$ iff $g(m)$ lies in the interval given by the interordinal attribute b.

Proposition 1 *(A,D) is a maximal bicluster of similar values ($A \subseteq G$, $D \subseteq M$) with the values lying in the interval $[t, t+\theta]$ for $t \in \mathbb{N}, \theta \geq 0$ iff (A,D,U) is a triadic concept of the context (G,M,B,Y) ,*

where $U = \{t+\theta, \dots q, \dots, q+t-1\}$. *Moreover, every triadic concept of the interordinally scaled triadic context* (G, M, B, Y) *is of the form* (A, D, U), *where* $A \subseteq G, D \subseteq M$, *and* $U = \{t+\theta, \dots q, \dots, q+t-1\}$ *for some* $t \in \mathbb{N}$ *and* $\theta \geq 0$.

Proof 1 *Let* (A, D) *be a maximal bicluster of similar values* ($A \subseteq G$, $D \subseteq M$), *then the values of attributes of the bicluster are lying in the interval* $[t, t+\theta]$ *for some* $t \in \mathbb{N}, \theta \geq 0$, *i.e.* $g(m) \in [t, t+\theta]$ *for every* $g \in A$, $m \in D$. *Due to the standard order on interordinal attributes, this implies that in the triadic context* (G, M, B, Y) *one has* $(g, m, b) \in Y$ *for all* $g \in A, m \in D$ *and* $b \in \{t+\theta, \dots q, \dots, q+t-1\}$ *and there is a rectangular parallelepiped* $(A, D, \{t+\theta, \dots q, \dots, q+t-1\})$ *filled with crosses in the triadic cross-table of* Y, *i.e.* $(A, D, \{t+\theta, \dots q, \dots, q+t-1\} \subseteq Y$. *This parallelepiped is inclusion-maximal, since otherwise this would mean that one can add either another object, or another attribute, or another scale value to its respective component. The possibility of adding another object or attribute would contradict the fact that* (A, D) *is a maximal bicluster, the possibility of adding another scale value would contradict the fact that the attribute values of the bicluster lie strictly in the interval* $[t, t+\theta]$. *Thus,* $(A, D, \{t+\theta, \dots q, \dots, q+t-1\})$ *is a triadic concept.*

In the opposite direction, consider a triadic concept (A, D, V) *in the interordinally scaled three-dimensional context, the attributes of* V *being ordered in the standard way. By the main property of the standard order on attributes of the interordinal scale, this would mean that for any two values* r *and* s *of* V, *the set* V *also contains all values in the interval* $[r, s]$. *Hence there are some* t *and* q *such that the values of* V *lie in the interval* $[t, t+\theta]$ *for all object-attribute pairs from* $A \times D$. *This means that* (A, D) *is a bicluster of similar values, which is maximal, since otherwise* (A, D, V) *would not have been a triadic concept.*

Example 10 $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \{t_3, t_4, t_5, t_6, t_7, t_8\})$ *is a triadic concept corresponding to the maximal bicluster* $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ *with* $\theta = 1$ *since* $\{t_3, t_4, t_5, t_6, t_7, t_8\}$ *is a modus characterizing interval* $[1, 2]$ *of length* 1.

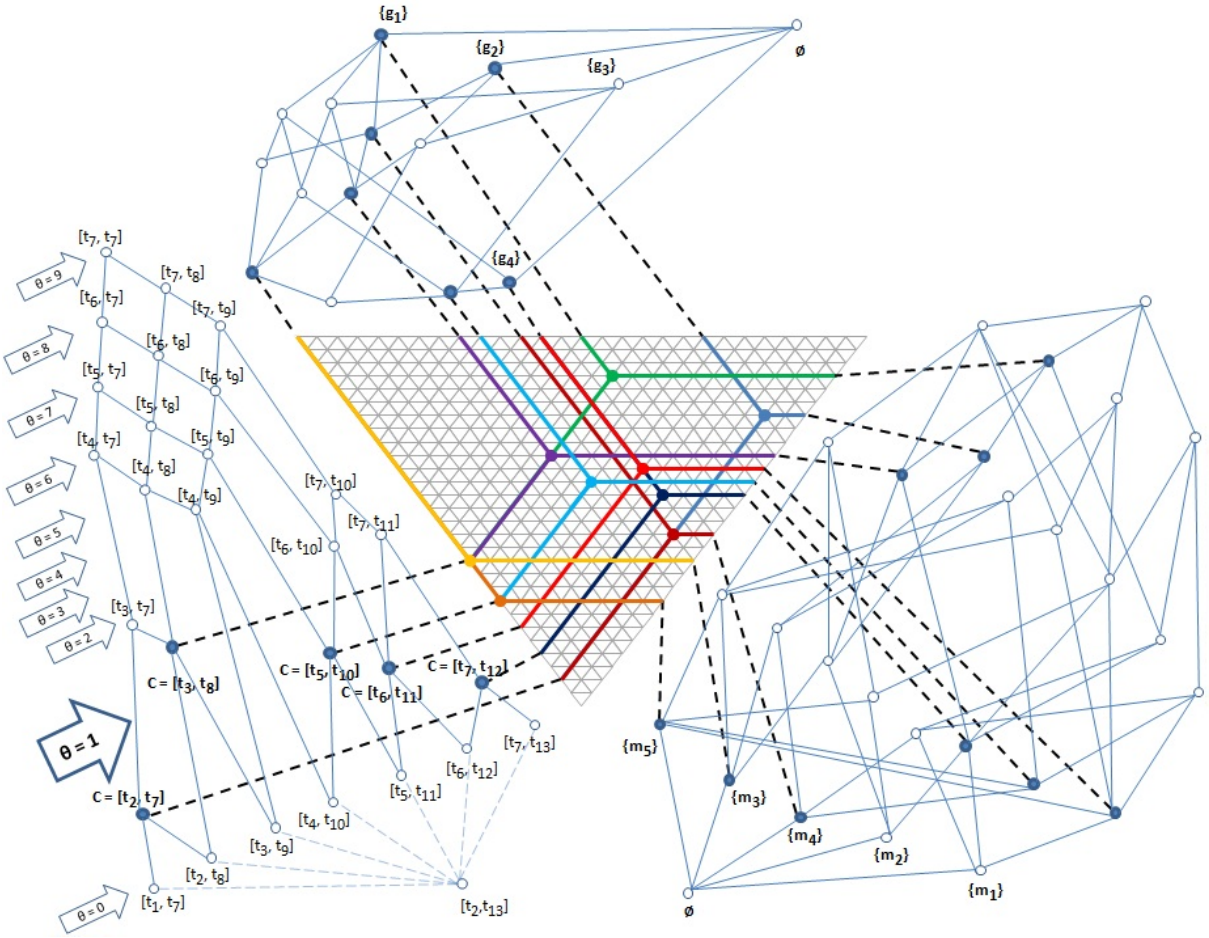


Figure 1.1: Trilattice from multi-valued context (Table 1.1) interordinally scaled to Table 1.5. Note, that only biclusters maximal for $\theta = 1$ are depicted. Consider the purple point that is the meet of the three purple lines: it represents the concept $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \{[t_3, t_8]\})$, i.e. with values in $[1, 2]$.

	$t_1 = [0,0]$					$t_2 = [0,1]$					$t_3 = [0,2]$					$t_4 = [0,6]$					$t_5 = [0,7]$				
	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5
g_1						×			×		×	×	×	×		×	×	×	×	×	×	×	×	×	×
g_2				×			×	×	×		×	×	×	×		×	×	×	×	×	×	×	×	×	×
g_3								×			×	×	×			×	×	×		×	×	×	×	×	×
g_4													×					×	×				×	×	×

	$t_6 = [0,8]$					$t_7 = [0,9]$					$t_8 = [1,9]$					$t_9 = [2,9]$					$t_{10} = [6,9]$				
	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5
g_1	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×	×		×					×
g_2	×	×	×	×	×	×	×	×	×	×	×	×	×		×	×				×					×
g_3	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×				×	×
g_4	×		×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		×	×

	$t_{11} = [7,9]$					$t_{12} = [8,9]$					$t_{13} = [9,9]$				
	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5	m_1	m_2	m_3	m_4	m_5
g_1															
g_2															
g_3				×											
g_4	×	×			×	×	×					×			

Table 1.5: Triadic scaled context from Table 1.1 with interordinal scaling. In gray lies the triadic concept $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\}, \{t_3, t_4, t_5, t_6, t_7, t_8\})$ corresponding to bicluster $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ with values in $[1, 2]$ and thus maximal for $\theta = 2 - 1 = 1$.

$G \supseteq A$ - extent	$M \supseteq B$ - intent	$T \supseteq C$ - modus	Interval over W
$A = \{g_1\}$	$B = \{m_1, m_2, m_3, m_4\}$	$C = [t_3, t_8]$	$[1, 2]$
$A = \{g_1, g_2\}$	$B = \{m_4\}$	$C = [t_2, t_7]$	$[0, 1]$
$A = \{g_1, g_2, g_3\}$	$B = \{m_1, m_2, m_3\}$	$C = [t_3, t_8]$	$[1, 2]$
$A = \{g_1, g_2, g_3, g_4\}$	$B = \{m_3\}$	$C = [t_3, t_8]$	$[1, 2]$
$A = \{g_1, g_2, g_3, g_4\}$	$B = \{m_5\}$	$C = [t_5, t_{10}]$	$[6, 7]$
$A = \{g_2\}$	$B = \{m_2, m_3, m_4\}$	$C = [t_2, t_7]$	$[0, 1]$
$A = \{g_3, g_4\}$	$B = \{m_4, m_5\}$	$C = [t_5, t_{10}]$	$[6, 7]$
$A = \{g_4\}$	$B = \{m_1, m_2\}$	$C = [t_7, t_{12}]$	$[8, 9]$
$A = \{g_4\}$	$B = \{m_1, m_5\}$	$C = [t_6, t_{11}]$	$[7, 8]$

Table 1.6: Triadic concepts with $\theta = 1$.

1.4.2 Trilattice diagram

In their seminal paper on TCA, Lehman and Wille proposed a way to visualize the ordered structure of triadic concepts [104]. This visualization possibility has not attracted a lot of attention since, hence we propose to illustrate it with derived triadic contexts from numerical data. Let us firstly recall notations of TCA: a triadic context is denoted by $\mathbb{K} = (K_1, K_2, K_3, Y)$, the set of all its triadic concepts by $\mathcal{I}(\mathbb{K})$ and its corresponding triadic diagram by $\underline{\mathcal{I}}(\mathbb{K})$.

Definition 13 (Quasi-order $\lesssim_i$ and equivalence relation $\sim_i$ on $\mathcal{I}(\mathbb{K})$) *Given two triadic concepts (A_1, A_2, A_3) and (B_1, B_2, B_3), three quasi-order and three equivalence are defined as follows, for $i = 1, 2, 3$*

$$(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3) \iff A_i \subseteq B_i, \tag{1.1}$$

$$(A_1, A_2, A_3) \sim_i (B_1, B_2, B_3) \iff A_i = B_i. \tag{1.2}$$

Definition 14 (Anti-ordinal dependencies) *A triadic concept is uniquely determined by two of its components since the three quasi-orders satisfy the anti-ordinal dependencies: For $\{i, j, k\} = \{1, 2, 3\}$, $(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3)$ and $(A_1, A_2, A_3) \lesssim_j (B_1, B_2, B_3)$ imply $(A_1, A_2, A_3) \gtrsim_k (B_1, B_2, B_3)$ for any two concepts (A_1, A_2, A_3) and (B_1, B_2, B_3).*

Definition 15 (Equivalence and factor sets) *For $i = 1, 2, 3$, the equivalence class of the relation $\sim_i$ which contains the concept (A_1, A_2, A_3) is denoted by $[(A_1, A_2, A_3)]_i$. $\lesssim_i$ induces an order $\leq_i$ on the factor set $\mathcal{I}(\mathbb{K})/\sim_i$: $[(A_1, A_2, A_3)]_i \leq_i [(B_1, B_2, B_3)]_i \iff A_i \subseteq B_i$. $(\mathcal{I}(\mathbb{K})/\sim_i, \leq_i)$ is the ordered set of all extents ($i = 1$), or intents ($i = 2$) and modi ($i = 3$) of $\mathbb{K}$.*

Definition 16 (Triadic diagram) *This relational structure $\underline{\mathcal{I}}(\mathbb{K})$ can be understood as two types of structures:*

- *The geometric structure: $(\mathcal{I}(\mathbb{K}), \sim_1, \sim_2, \sim_3)$: It is represented as a partial 3-net, i.e. a triangular pattern. The three equivalence relations are here represented by 3 systems of parallel lines. For example, consider the equivalence relation on concepts with $i = 1$: concepts of an equivalence class have same extent and are depicted on the same line. As such, the classes of equivalence meet at most in one element for a given concept.*
- *The ordered structures: $(\mathcal{I}(\mathbb{K})/\sim_i, \leq_i)$: Each of them is represented by a Hasse diagram.*

Figure 1.1 presents the trilattice obtained from our running example (i.e. Table 1.5). For sake of readability, we highlight there only the biclusters that are maximal for $\theta = 1$. Taking the concept $(\{g_4\}, \{m_1, m_5\}, [6, 11])$ from the Table 1.6, the three (pairwise non parallel) lines, corresponding respectively to the equivalence class of the extent $\{g_4\}$, the intent $\{m_1, m_5\}$ and the modus $[6, 11]$, only meet in one point of the triangular pattern which represents this concept. The three quasi-order structures of extents, intents and modi, i.e. Hasse diagrams of all $(\mathcal{I}(\mathbb{K})/\sim_i, \leq_i)$, lie around the trilattice.

1.4.3 Handling n-ary numerical dataset

A straightforward generalization of the presented approach lies in its potential extension to n-ary numerical datasets. The basic idea is as follows. Consider a numerical dataset with n dimensions, e.g. *genes* $\times$ *biological situations* $\times$ *timestamps* when $n = 3$. Then, one can extract n-clusters of similar values by scaling the numerical data into a $n + 1$-dimensional binary dataset. So-called polyadic concepts [162] in the binary dataset are here again in 1-to-1-correspondence with maximal n-clusters of similar values of the numerical dataset. We present here theoretical aspects while computing aspects can be regarded with the existing algorithms Data-Peeler [43].

Recall that the *standard order* on $2k-1$ attributes of the interordinal scale is as follows: $[v_1, v_1], [v_1, v_2]$, $\ldots, [v_1, v_k], [v_2, v_k] \ldots, [v_k, v_k]$. Having the standard order on the attributes of the interordinal scale one can enumerate them from 1 to $2k+1$. Let $(G_1, \ldots, G_n, W, I)$ be an n-dimensional many-valued context, i.e., an $n + 1$-dimensional relation $I \subseteq G_1 \times \ldots \times G_n \times W$ and W ($|W| = q$) be the set of numerical values enumerated in the ascending order from 1 to q, and let $v(g_1, \ldots, g_n)$ be a map taking the tuple $g_1, \ldots, g_n$ to the value $w \in W$. Let the numerical values from W be interordinally scaled with the standard order on the scale attributes, so we can denote the scale attributes by $m_1, \ldots, m_q, \ldots, m_{2q-1}$. Let $B = \{m_1, \ldots, m_q, \ldots m_{2q-1}\}$ and $Y \subseteq G_1 \times \ldots \times G_n \times B$ be an $n + 1$-ary relation such that $(g_1, \ldots, g_n, m) \in Y$ iff the value w of the n-tuple $g_1, \ldots, g_n$ lies in the interval given by the interordinal attribute m.

Proposition 2 *$(A_1, \ldots, A_n)$ is a maximal n-way cluster of similar values ($A_i \subseteq G_i$) with the values lying in the interval $[t, t+\theta]$ for $t \in \mathbb{N}, \theta \geq 0$ iff $(A_1, \ldots, A_n, U)$ is an $n+1$-adic concept of the $n+1$-dimensional context $(G_1, \ldots, G_n, U, Y)$, where $U = \{t+\theta, \ldots q, \ldots, q+t-1\}$. Moreover, every $n+1$-dimensional concept of the interordinally scaled $n + 1$-dimensional context $(G_1, \ldots, G_n, W, Y)$ is of the form $(A_1, \ldots, A_n, U)$, where $A_i \subseteq G_i$, and $U = \{t + \theta, \ldots q, \ldots, q + t - 1\}$ for some $t \in \mathbb{N}$ and $\theta \geq 0$.*

The proof is similar as in the triadic case and hence is omitted.

1.4.4 Remarks

We showed that extracting biclusters of similar values for any θ in a numerical dataset can be achieved by (i) scaling the attribute value dimension and (ii) extracting the triadic concepts in the resulting derived triadic context. The same applies when considering n-ary numerical datasets.

On the one hand, triadic concepts (A, B, U) with the largest sets A, B or C represent large biclusters of similar values. Indeed, the larger $|A|$ and $|B|$ the larger the data covering of the corresponding bicluster. Furthermore, the larger $|U|$, the more similar values for bicluster (A, B). Indeed, by the properties of interordinal scaling, the more intervals in U, the smaller their interval intersection. Mining so-called top-k frequent triadic concepts can accordingly be achieved with the existing algorithm Data-Peeler [43].

On the other hand, extracting maximal biclusters for all θ may be neither efficient nor effective with large numerical data: their number tends to be very large and not all biclusters are relevant for a given analysis. Furthermore, both size and density of contexts derived with interordinal scaling are known to be problematic w.r.t algorithmic scalability, see e.g. [91]. In existing methods of the literature, θ is set *a priori*. We show now how to handle this case with slight modifications, this is our second main result.

1.5 Extracting Biclusters of Similar Values For a Given θ

In this section, we present our second contribution. We consider the problem of extracting maximal biclusters of similar values in TCA for a given θ only. It comes with slight modifications of the methodology presented in the previous section, but requires more algorithmic considerations: although all triadic concepts correspond to biclusters of similar values with a new transformation procedure, it is not sure that such concepts correspond to maximal biclusters. In this way, it is not possible to use concepts extraction

algorithms directly (or it would require post-processing which is always a solution to avoid). Accordingly, a modified scaling procedure will lead us to the design of the algorithm TriMax for a complete and correct extraction of maximal biclusters for a given θ. Finally, we experiment with this new algorithm.

1.5.1 Scaling numerical data in a triadic context

Consider the previous scaling applied to a numerical dataset (G, M, W, I). It scales W into a dimension T and all subsets of T characterize all intervals of values over W. To get maximal biclusters for a given θ only, we should not consider all possible intervals in W, but rather all intervals (i) having a range size that is less or equal than θ to avoid biclusters with non similar values, and (ii) having a range size the closest as possible to θ to avoid non-maximal biclusters. For example, if we set $\theta = 2$, it is probably not interesting to consider interval $[0, 8]$ in the scale dimension since $8 - 0 > \theta$. Similarly, considering the interval $[6, 6]$ may not be interesting as well, since a bicluster with all its values equal to 6 may not be maximal. As introduced in [86], the maximal intervals of similar values used for the scale are called *blocks of tolerance* over the set of numbers W with respect to the tolerance relation $\simeq_\theta$. We now recall basics on tolerance relations over a set of numbers. This allows us to define a simpler scaling procedure. The resulting triadic context is then mined with a new TCA algorithm called TriMax to extract maximal biclusters of similar values for a given θ. Blocks of tolerance over W are as maximal sets of pairwise similar values:

Definition 17 (Tolerance relation and blocks) *A binary relation $\simeq$ is a tolerance if it is reflexive, symmetric but not necessarily transitive. Given a set W, a subset $V \subseteq W$, and a tolerance relation $\simeq$ over W, V is a* block of tolerance *if:*
(i) $\forall w_1, w_2 \in V,\ w_1 \simeq w_2$ (pairwise similarity)
(ii) $\forall w_1 \notin V, \exists w_2 \in V,\ w_1 \not\simeq w_2$ (maximality).

It follows that $\simeq_\theta$ is a tolerance relation. From Table 1.1 we have $W = \{0, 1, 2, 6, 7, 8, 9\}$. With $\theta = 2$, one has $0 \simeq_2 2$ but $2 \not\simeq_2 6$. Accordingly, one obtains 3 blocks of tolerance, namely the sets $\{0, 1, 2\}$, $\{6, 7, 8\}$ and $\{7, 8, 9\}$. These three sets can be renamed as the convex hull of their elements on $\mathbb{N}$: respectively, $[0, 2]$, $[6, 8]$ and $[7, 9]$: any number lying between the minimal and the maximal elements (w.r.t. natural number ordering) of a block of tolerance is naturally similar to any other element of the block. Then, to derive a triadic context from a numerical dataset, we simply use tolerance blocks over W to define the scale dimension.

Definition 18 (TriMax scale relation) *The scale relation is a binary relation $J \subseteq W \times C$, where C is the set of blocks of tolerance over W renamed as their convex hulls. Then, $(w, c) \in J$ iff $w \in c$.*

Example 11 *From Table 1.1 we have: $C = \{[0, 1], [1, 2], [6, 7], [7, 8], [8, 9]\}$ with $\theta = 1$, and $C = \{[0, 2], [6, 8],\ [7, 9]\}$ with $\theta = 2$.*

In this way, we can apply the same context derivation as in the previous section: scaling is still based on intervals, but this time it uses tolerance blocks.

Definition 19 (TriMax triadic scaled context) *Let $Y \subseteq G \times M \times C$ be a ternary relation. Then $(g, m, c) \in Y$ iff $(m(g), c) \in J$, or simply $m(g) \in c$, where J is the scale relation. (G, M, C, Y) is called the TriMax triadic scaled context.*

Example 12 *Table 1.7 is the TriMax triadic scaled context derived from the numerical dataset lying in Table 1.1 with $\theta = 1$.*

Definition 20 (Dyadic context associated with a block of tolerance) *Consider a block of tolerance $c \in C$. The dyadic context associated with this block is given by (G, M, Z) where Z denotes the set of all $(g, m) \in G \times M$ such that $m(g) \in c$.*

	[0,1]				[1,2]				[6,7]		[7,8]			[8,9]	
	m_1	m_2	m_3	m_4	m_1	m_2	m_3	m_4	m_4	m_5	m_1	m_4	m_5	m_1	m_2
g_1	×			×	×	×	×	×		×					
g_2		×	×	×	×	×	×			×					
g_3			×		×	×	×		×	×		×			
g_4							×		×	×	×		×	×	×

Table 1.7: Triadic scaled context using tolerance blocks over W and $\theta = 1$ (empty columns are not displayed). The bicluster $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ with values in $[1, 2]$ and maximal for $\theta = 1$ corresponds to a dyadic concept in the dyadic context labeled $[1, 2]$.

Example 13 *In Table 1.7, each dyadic context is labeled by its corresponding block of tolerance for $\theta = 1$.*

Now, note that blocks of tolerance over W are totally ordered: let $[v_1, v_2]$ and $[w_1, w_2]$ be two blocks of tolerance, one has $[v_1, v_2] < [w_1, w_2]$ iff $v_1 < w_1$. Hence, associated dyadic contexts are also totally ordered and we can use an indexing set to label them (as done in the algorithm pseudo-code later).

We now present our next results: the scaled triadic context supports the extraction of maximal biclusters of similar values for a given θ. In this case however, existing algorithms of TCA cannot be applied directly. For example, in Table 1.7, the triadic concept $(\{g_3\}, \{m_4\}, \{[6, 7], [7, 8]\})$ corresponds to a bicluster of similar values which is not maximal. Hence we present hereafter a new TCA algorithm for this task, called TriMax.

The basic idea of TriMax relies on the following facts. Firstly, since each dyadic context corresponds to a block of tolerance, we do not need to compute intersections of contexts, such as classically done in TCA. Hence each dyadic context is processed separately. Secondly, a dyadic concept of a dyadic context necessarily represents a bicluster of similar values, but we cannot be sure it is maximal (see previous example). Hence, we need to check if a concept is still a concept in other dyadic contexts, corresponding to other classes of tolerance. This is made precise with the following proposition.

Proposition 3 *Let (A, B, U) be a triadic concept from* TriMax *triadic scaled context (G, M, C, Y), such that U is the outer closure of a singleton $\{c\} \subseteq C$. If $|U| = 1$, (A, B) is a maximal bicluster of similar values. Otherwise, (A, B) is a maximal bicluster of similar values iff there is no $\in [min(U); max(U)]$, $y < c$ such that $(A, B) \neq \Psi_y^{'}(\Psi_y((A, B)))$, where $\Psi_y^{'}(\cdot)$ and $\Psi_y(\cdot)$ correspond to inner derivation operators associated with y^{th} dyadic context.*

Proof 2 *When $|U| = 1$, (A, B) is a dyadic concept only in one dyadic context corresponding to a block of tolerance. By properties of tolerance blocks, (A, B) is a maximal bicluster. If $|U| \neq 1$, (A, B) is a dyadic concept in $|U|$ dyadic contexts. Since the tolerance block set is totally ordered, it directly implies that modus U is the interval $[min(U); max(U)]$. Hence, if there is $y \in [min(U); max(U)]$ such that $(A, B) = \Psi_y^{'}(\Psi_y((A, B)))$, then (A, B) is not a maximal bicluster of similar values.*

1.5.2 The TriMax algorithm

The TriMax algorithm starts with scaling initial numerical data into several dyadic contexts, each one standing for a block of tolerance over W with given θ. The set of all dyadic contexts forms accordingly a triadic context. Then, each dyadic context is mined with any FCA algorithm (or closed itemset mining algorithm), and all formal concepts are extracted. For a given concept (A, B), we compute outer derivation $\Phi^{'}((A, B))$, i.e. to obtain the set of dyadic contexts labels in which the current dyadic concept holds. If this set is a singleton, this means that (A, B) is a concept for the current block of tolerance only, i.e. it is a maximal bicluster of similar values, and it has been, or will never be, generated twice. Otherwise, (A, B) is a concept in other contexts, and can be generated accordingly several times (as much as the number of contexts in which it holds). Then, we only consider (A, B) if we are sure it is the last time it is computed. Finally, we need to check if current concept represents a maximal bicluster, i.e. there should not exist a context labeled by an element of the modus where (A, B) is not a dyadic concept.

Algorithm 1 TriMax

Given: Numerical dataset (G, M, W, I), tolerance parameter θ
Output: Maximal biclusters of similar values
Let $C = \{[a_i, b_i]\}$ be the totally ordered set of all blocks over W for given θ. Indices i form an indexing set.
for $[a_i, b_i] \in C$ **do** Build context (G, M, Z_i) such that $(g, m) \in Z_i \Leftrightarrow m(g) \in [a_i, b_i]$
end for
for (G, M, Z_i) **do**
    Use any FCA algorithm to extract all its concepts (A, B)
    for dyadic concepts (A, B) in the current context (G, M, Z_i) **do**
        if $|\Phi^{'}((A, B))| = 1$ **then**
            print (A, B)
        else
            if $max(\Phi^{'}((A, B)) = i$ **then**
                $x \leftarrow min(\Phi^{'}((A, B))$
                if $\nexists y \in [x, i[$ s.t. $(A, B) \neq \Psi_y^{'}(\Psi_y((A, B)))$ **then**
                    print (A, B)
                end if
            end if
        end if
    end for
end for

Proposition 4 TriMax *outputs a (i) complete, (ii) correct and (iii) non redundant collection of all maximal biclusters of similar values for a given numerical dataset and similarity parameter θ.*

Proof 3 *(i) and (ii) follow directly from Proposition 3. Statement (iii) is ensured by the second if condition of the algorithm: a dyadic concept (or equivalently bicluster) is considered iff it has been extracted in the last dyadic context in which it holds.*

We do not report experiments in this document but they can be found in [88]. We also successfully compared with the two methods we found in the literature that also consider the problem of extracting all maximal biclusters of similar values from a numerical dataset. The first method is called *Numerical Biset Miner* (NBS-Miner [25]). The second method is one we designed with pattern structures during my PhD (IPS [89]).

1.6 Conclusion

We addressed the problem of biclustering numerical data with Formal Concept Analysis. (Maximal) biclusters of similar values can be characterized and extracted with Triadic Concept Analysis, which turns out to be a novel mathematical framework for this task. We have defined a scaling procedure turning original numerical data into triadic contexts from which biclusters can be extracted as triadic concepts with existing algorithms. This approach allows a correct, complete and non-redundant extraction of all maximal biclusters, for any similarity parameter θ and can be extended to n-ary numerical datasets while their computation can be directly distributed. The interpretation of triadic concepts is powerful: both extent and intent allow one to characterize a bicluster (i.e. the rectangle), while the modus gives the range of values of the biclusters, and for which θ is the bicluster maximal. Moreover, the larger the modus, the more similar the values within a current bicluster. This fact gives a particular semantics to the notion of *support* as defined in itemset-mining [8]. We also adapted the TCA machinery with algorithm TriMax to extract maximal biclusters for a user-defined threshold θ. It appears that TriMax is a fully customizable algorithm: any concept extraction algorithm (or pattern mining algorithm) can be used as a core module (along with several constraints on produced dyadic concepts), while its distributed computation is direct.

Chapter 2

Database Dependency Discovery

This chapter highlights links between the discovery of data dependencies in the Database field and the discovery of implications of a well defined formal context on one hand, and of a pattern structure on the other, in the field of FCA. We consider in this chapter only functional and degenerated multivalued dependencies [20], although we conclude that other types of dependencies can be generically formalized. As a sequel indeed, we presented how Approximate-Matching Dependencies and Ordered Dependencies, can also be considered with slight changes [18, 46].

The chapter is organized as follows. Basics on functional dependencies are presented in Section 2.2. Then, how FCA can be used to characterize functional dependencies after a costly data transformation is described in Section 2.3. It is followed by our main contribution which consists in characterizing functional dependencies with pattern structures (Section 2.4). Section 2.5 handles the case of degenerated multivalued dependencies.

2.1 Introduction

The discovery of functional dependencies is an important topic in the database field since they represent the fact that the value of one or several attributes is uniquely (functionally) determined by the values of other attributes. As such, they are valuable in order to explain the normalization of a database schema in the Relational Database Model. Consider the relation $AddressBook(id, name, street, ZIP, City)$: it entails the functional dependencies stating that any two tuples of this relation that have the same value of ZIP code, also have the same value for the attribute $City$. Formally, given a relation schema $\mathcal{U}$, i.e. a set of attributes to describe some objects or tuples, a functional dependency is denoted by $X \to Y$, $X, Y \subseteq \mathcal{U}$ and means that the objects that take the same values for the attributes in X take also the same values for the attributes in Y. Table 2.1 in Section 2.2 is a tabular representation of a relation. Rows denote objects (or tuples) and columns denote attributes of the schema. There, the functional dependency $a \to d$ holds: when t_1 and t_3 take the same value for the attribute a, they also take a same value for the attribute d. In the relational database model there are different types of dependencies (conditional [56], impurity [146], DMVDs [143], etc., see [85] for a more detailed survey), although functional dependencies are among the most popular, and have been widely studied [1, 147, 112, 139, 155].

Besides, functional dependencies, and dependencies in general, are closely linked to attribute implications in Formal Concept Analysis [64]. FCA is an important mathematical framework rooted in lattice theory that is also used for data-analysis purposes (deeply described in [64]). Among other, it aims at discovering implicit relations between objects and their attributes. It starts with a triple (G, M, I), called a formal context, where G is a set of objects, M a set of attributes and I a binary relation such as $I \subseteq G \times M$. So called implications are expressions of the form $X \to Y$, $X, Y \subseteq M$ stating that when an object has attributes in X, then it has also attributes in Y.

As such, functional dependencies (FDs) and attribute implications are expressions of the same form,

i.e. $X \to Y$, defined over a set of attributes. However, in the first case, FDs are defined on numerical or categorical attributes, while implications are defined on binary attributes. Thus, to show an equivalence (or just links) between FDs and implications, the original data in which FDs hold have to be transformed into a formal context, whose implications can then be compared to FDs. This was actually presented in the book of FCA (see [64], page 92) and as well in [108]. It was shown how to build a formal context from the original data and that the implications in this formal context are syntactically equivalent to the FDs of the original data. The second table in Figure 2.1 shows the formal context obtained from the original data in Table 2.1: whereas the procedure is explained later, one should notice that indeed the implication $a \to d$ holds, which is also a FD in the original data. Unfortunately, the number of objects of the resulting context is quadratic w.r.t. the original, which does not allow this method to be applied on large datasets.

The previous remark is actually the motivation of the present work leading to the following question: *Can we characterize with FCA functional dependencies as implications, avoiding a significantly larger data representation?* We positively answer this question by introducing a method based on Pattern Structures [62]. A pattern structure can be understood as a generalization of standard FCA to handle complex data (say, non binary): instead of a binary relation between some objects and their attributes, it applies on a relation between objects and their descriptions that form a particular partially ordered set. Our approach consists in considering that the attributes from the original relation schema $\mathcal{U}$ can be described by a partition over the set of tuples, and that the set of partitions forms a lattice. As such, so-called *partition pattern structures* are introduced in this paper, and we show that the implications they hold are equivalent to the functional dependencies, as well as the attribute implications holding in the formal context introduced in the previous section.

Consequently, our contribution is three-fold:

- Firstly, we present a new conceptual structure, called partition pattern structure.
- Secondly, we show how such a structure can be built from a numerical dataset to characterize functional dependencies: The interest is to prove that pattern structures are a flexible mechanism within FCA to encode the semantics of the dependencies without a heavy data representation.
- Finally, we show that this method allows one, with a minor variation, to characterize another kind of dependencies called degenerated multi-valued dependencies (DMVDs, introduced later). We also propose experiments showing that our conceptual structure has better computational properties than the classical FCA approach.

2.2 Functional and Degenerated Multivalued Dependencies

We first introduce functional dependencies (FDs). Let $\mathcal{U}$ be a set of attributes, and let *Dom* be a set of values (a domain). For sake of simplicity, we assume that *Dom* is a numerical set. A tuple t is a function $t : \mathcal{U} \mapsto Dom$, and a table T is a set of tuples. Usually tables are presented as a matrix, as in Table 2.1, where the set of tuples (or objects) is $T = \{ t_1, t_2, t_3, t_4 \}$ and $\mathcal{U} = \{ a, b, c, d \}$ is the set of attributes. We use *table, dataset, set of tuples* as equivalent terms. We overload the functional notation of a tuple in such a way that, given a tuple $t \in T$, we say that $t(X)$ (for all $X \subseteq \mathcal{U}$) is a tuple with the values of t in the attributes $x_i \in X$:

$$t(X) = \langle t(x_1), t(x_2), \ldots, t(x_n) \rangle$$

For example, we have that $t_2(\{ a, c \}) = \langle t_2(a), t_2(c) \rangle = \langle 4, 4 \rangle$. In the paper, the set notation is dropped: instead of $\{ a, b \}$ we use ab.

Definition 21 ([155]) *Let T be a set of tuples, and $X, Y \subseteq \mathcal{U}$. A* **functional dependency (FD)** *$X \to Y$ holds in T if:*

$$\forall t, t' \in T : t(X) = t'(X) \implies t(Y) = t'(Y)$$

id	a	b	c	d
t_1	1	3	4	1
t_2	4	3	4	3
t_3	1	8	4	1
t_4	4	3	7	3

Table 2.1: An example of a table T, i.e. a set of tuples

For instance, the functional dependencies $a \to d$ and $d \to a$ hold in T, whereas the functional dependency $a \to c$ does not hold since $t_2(a) = t_4(a)$ but $t_2(c) \neq t_4(c)$.

We now present a generalization of functional dependencies: degenerated multivalued dependencies.

Definition 22 ([143]) *Let $X, Y, Z \subseteq \mathcal{U}$ of a table T, such that $X \cap Y = X \cap Z = Y \cap Z = \emptyset$ and $X \cup Y \cup Z = \mathcal{U}$. We say that a* ***degenerated multivalued dependency (DMVD)*** *$X \to Y$ holds in T if and only if:*

$$\forall t, t' \in T : t(X) = t'(X) \implies t(Y) = t'(Y) \text{ or } t(\mathcal{U} \setminus X \setminus Y) = t'(\mathcal{U} \setminus X \setminus Y)$$

For instance, we have that $a \to b$ holds in the example table T, since $t_1(a) = t_3(a)$ and $t_1(cd) = t_3(cd)$, and $t_2(a) = t_4(a)$ and $t_2(b) = t_4(b)$. We remark that the functional dependency $a \to b$ does not hold in T, because of the pair of tuples t_1, t_3. Degenerated multivalued dependencies are a generalization of functional dependencies: if we drop the clause $t(\mathcal{U} \setminus X \setminus Y) = t'(\mathcal{U} \setminus X \setminus Y)$, we have the definition of functional dependencies. Therefore, if the functional dependency $X \to Y$ holds, then, the degenerated multivalued dependencies $X \to Y$ and $X \to \mathcal{U} \setminus X \setminus Y$ hold as well, whereas the opposite is not necessarily true, as the previous example shows.

Dependencies have a set of axioms stating which dependencies hold given an arbitrary set of dependencies of the same kind. The set of dependencies Σ closed under their own set of axioms is denoted by Σ^+. A minimal set of dependencies from which all other dependencies can be deduced by means of those axioms is called a *minimal generating set*.

Let $\mathcal{U}$ be the set of attributes of a relational table. The axioms for functional dependencies follow Armstrong rules [155] for all $X, Y, Z \subseteq \mathcal{U}$:

$$\frac{Y \subseteq X}{X \to Y} \quad \frac{X \to Y}{X \cup Z \to Y} \quad \frac{X \to Y, Y \to Z}{X \to Z}$$

These axioms are respectively called reflexivity, augmentation and transitivity. Implications also share the same axioms [64]. On the other hand, the axioms for degenerated multivalued dependencies (DMVDs) are reflexivity, complementation or symmetry, augmentation and transitivity, i.e. for all $X, Y, Z, V, W \subseteq \mathcal{U}$:

$$\frac{Y \subseteq X}{X \to Y} \quad \frac{X \to Y}{X \to \mathcal{U} \setminus Y \setminus X} \quad \frac{X \to Y, V \subseteq W}{W \cup X \to V \cup Y} \quad \frac{X \to Y, Y \to Z}{X \to Z \setminus Y}$$

These axioms are also shared by multivalued dependencies, a well-known kind of dependencies in the relational database model [143].

In this paper, we will use pattern structures to characterize a set of functional dependencies (and DMVDs) that hold in data table. In fact, it is not an arbitrary Σ, but precisely Σ^+. Firstly, we recall how such characterization is classically done in the FCA literature.

2.3 Characterizing Functional Dependencies with FCA

2.3.1 Functional Dependencies as Implications

We now recall with an example how functional dependencies can be characterized using FCA (see [17] and [64], page 92). The main idea behind this method consists in transforming a man-valued context into a formal context, whose concept lattice characterizes functional dependencies.

Starting from a tuple table T with attributes $\mathcal{U}$ taking values in Dom, we build the formal context $\mathbb{K} = (\mathcal{B}_2(G), M, I)$, where $G = T$ and $M = \mathcal{U}$ to respect the FCA notations from [64]. $\mathcal{B}_2(G) = \{ (t_i, t_j) \mid i < j \text{ and } t_i, t_j \in T \}$ is the set of pairs of tuples from G. Then, the relation I is defined as

$$(t_i, t_j)\ I\ m \Leftrightarrow t_i(m) = t_j(m), \text{ for } m \in M$$

This binary relation between pairs of tuples and attributes is reflexive, symmetric and transitive, and, therefore, it is an *equivalence relation*. The objects of $\mathbb{K}$ correspond to the set of all pairs of tuples from T (excluding symmetry and reflexivity to avoid redundancy), while attributes remain the same. $((t_i, t_j), m) \in I$ means that the tuples t_i and t_j agree on the value taken by the attribute $m \in M$. Figure 2.1 illustrates the transformation of the initial data to build a formal context and its concept lattice. It should be noticed that the number of objects of the formal context is in the range of $O(|T^2|)$ (where $|T|$ is the number of tuples), so it can be significantly larger than the original set of tuples T.

id	a	b	c	d
t_1	1	3	4	1
t_2	4	3	4	3
t_3	1	8	4	1
t_4	4	3	7	3

$\mathbb{K}$	a	b	c	d
(t_1, t_2)		×	×	
(t_1, t_3)	×		×	×
(t_1, t_4)		×		
(t_2, t_3)			×	
(t_2, t_4)	×	×		×
(t_3, t_4)				

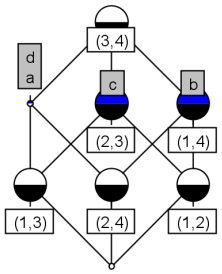


Figure 2.1: Characterizing FDs with FCA: from a set of tuples to a formal context and its concept lattice.

We now explain how this concept lattice characterizes the set of all functional dependencies that hold in the table T with the following proposition:

Proposition 5 ([64, 17]) *A functional dependency $X \to Y$ holds in a table T if and only if $\{X\}'' = \{X, Y\}''$ in the formal context $\mathbb{K} = (\mathcal{B}_2(G), M, I)$.*

This proposition states how to test that a FD holds using the concept lattice that has been computed. For instance, let us suppose that we want to test whether a functional dependency $a \to b$ holds in the formal context of Figure 2.1. We should test in the corresponding concept lattice if $\{a\}'' = \{a, b\}''$. In this particular case, we have that $\{a\}'' = \{a, d\}$ and $\{a, b\}'' = \{a, b, d\}$, which means that this dependency does not hold in T. On the other hand, the dependency $ac \to d$ holds, since $\{a, c\}'' = \{a, c, d\}$ and $\{a, c, d\}'' = \{a, c, d\}$.

An interesting consequence is that the set of implications that hold in the formal context $\mathbb{K} = (\mathcal{B}_2(G), M, I)$ is syntactically equivalent to the set of functional dependencies that hold in a table T [64, 17]. By *syntactically* we mean that whenever an implication $X \to Y$ holds in $\mathbb{K}$, then the functional dependency $X \to Y$ holds in T (though not left-reduced). Equivalently, the minimal generating set of functional dependencies that hold in T is the same as the Duquenne-Guigues basis of the implications that hold in $\mathbb{K}$. Going back to our example, the concept lattice given in Figure 2.1 characterizes the implications $a \to d$ and $d \to a$, which form the Duquenne-Guigues basis.

2.3.2 Conceptual Scaling and FDs

Before introducing our method based on pattern structures to characterize functional dependencies, we investigate another aspect of the original data transformation into a formal context. In FCA, a way to turn a numerical table into a formal context is to use a conceptual scale (see Chapter 1.3 of [64]). Conceptual scaling consists in turning the many-valued attributes into binary attributes following rules given by the scale. For example, the ordinal scale states that, for a numerical attribute m, a pair object-attribute (g, m) taking value $x \in \mathbb{N}$ should be derived into binary attributes "$\leq y$", for any $y \geq x$ of the attribute domain, i.e. $(g, \text{“} \leq y\text{”}) \in I$. This means that the original dataset is turned into a formal context having the same set of objects and a larger set of binary attributes.

In the previous subsection, the data transformation we presented is not a conceptual scaling: the set of attributes remains the same after the transformation, whereas the set of objects is changed and its size is increased. Indeed, we replace objects by pairs of objects, and, given n objects, there are $n(n-1)/2$ potential pairs of objects. By contrast, we investigate in this section whether, given a data table T, it is possible to define a conceptual scaling applied to attributes and allowing to derive a formal context $\mathcal{K}_T$ with the same set of objects and such that the set of FDs holding in T is syntactically equivalent to the set of attribute implications holding in $\mathcal{K}_T$.

We show in the following example that this is not possible by constructing a simple and suitable counter-example. Let us consider the $n \times m$ numerical data table given in Figure 2.2 (left), based on $n = 4$ rows (objects) and $m = 4$ columns (attributes). Here the fact that $n = m$ here does not affect generality. The binarization, i.e. the transformation applied to objects (that could be termed as "vertical scaling"), yields $n(n-1)/2 = 6$ rows. The singularity of this example is that for any attribute, all objects share the same value except one (no empty row in the binary table), and this particular object is different for each attribute.

Actually, the context in Figure 2.2 (middle) is "clarified" and "reduced". Recall that a formal context (G, M, I) is clarified if $\forall g, h \in G$, $g' = h'$ implies $g = h$ (and similarly for the attributes). Moreover, an element x in a lattice L is $\vee$-irreducible (resp. $\wedge$-irreducible) if $x \neq \bot$ (resp. $x \neq \top$) and $x = a \vee b$ (resp. $x = a \wedge b$) implies $x = a$ or $x = b$ for all $a, b \in L$ [47]. Then in terms of FCA, a clarified context (G, M, I) is reduced when it is row-reduced (i.e. every object-concept is $\vee$-irreducible) and column-reduced (i.e. every attribute-concept is $\wedge$-irreducible) [64]. In addition, the number jir of $\vee$-irreducible concepts in a concept lattice is less than or equal to the number of objects $|G|$, and the number mir of $\wedge$-irreducible concepts is less than or equal to the number of attributes. There is equality when the formal context is clarified and reduced. For example, for the context given in Figure 2.2 (middle) and the associated concept lattice given in Figure 2.2 (right), we can observe that $mir = 4$ and $jir = 6$.

id	a	b	c	d
t_1	1	2	3	1
t_2	1	2	1	4
t_3	1	1	3	4
t_4	2	2	3	4

id	a	b	c	d
(t_1, t_2)	x	x		
(t_1, t_3)	x		x	
(t_1, t_4)		x	x	
(t_2, t_3)	x			x
(t_2, t_4)		x		x
(t_3, t_4)			x	x

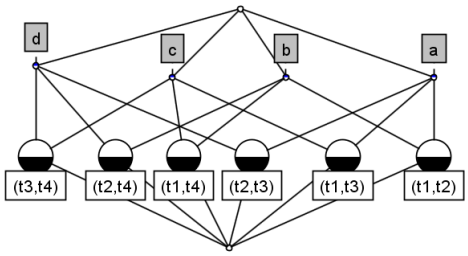


Figure 2.2: A data table T (left) with its associated formal context $(\mathcal{B}_2(G), M, I)$ (middle). In the concept lattice diagram, nodes labeled with attributes (upper level) are $\wedge$-irreducible concepts while nodes labeled with objects (lower level) are $\vee$-irreducible concepts (right).

Now, scaling the data table T in Figure 2.2 (left) keeping unchanged the set of objects $G = T$ returns a formal context, say $(G, \hat{M}, \hat{I})$, where $|\hat{M}| \geq |M|$, i.e. the number of scaled attributes is greater than or equal to the initial number of attributes. Then, the number of $\wedge$-irreducible elements mir in the concept lattice $\underline{\mathfrak{B}}(G, \hat{M}, \hat{I})$ should verify $mir \geq 4$, as scaling separates attributes rather than merging them. In

the same way, the number of $\vee$-irreducible elements jir in $\underline{\mathfrak{B}}(G, \hat{M}, \hat{I})$ should verify $jir \leq 4$. By contrast, $mir = 4$ and $jir = 6$ for the lattice $\underline{\mathfrak{B}}(\mathcal{B}_2(G), M, I)$. Then, it is not possible to find any scaling yielding a concept lattice $\underline{\mathfrak{B}}(G, \hat{M}, \hat{I})$ isomorphic to $\underline{\mathfrak{B}}(\mathcal{B}_2(G), M, I)$ –and thus with the same implication basis– as $\vee$-irreducible and $\wedge$-irreducible elements are preserved by the isomorphism. Thus, binarization should be necessarily applied to objects and not to attributes.

2.4 Characterizing FDs with Pattern Structures

In the previous section, we showed how to turn a set of tuples T into a formal context $\mathbb{K} = (\mathcal{B}_2(G), M, I)$, whose concept lattice allows to characterize functional dependencies. However, the number of objects $|\mathcal{B}_2(G)|$ in the resulting context is quadratic with respect to the number of tuples. As shown later in the experiments, this is not viable for real datasets. Thus, we propose to use the formalism of pattern structures to obtain an equivalent concept lattice, avoiding a transformation leading to a quadratic number of objects. Pattern structures can be understood as a generalization of FCA able to directly deal with complex data i.e. objects taking descriptions in a partially ordered set.

2.4.1 The Partition Lattice as a Space of Descriptions

In order to construct the meet-semi-lattice of potential object descriptions of a pattern structure, we recall well-known definitions of the partitions of a set and the so-called partition lattice. In the examples that follow, we consider a set $E = \{1, 2, 3, 4\}$.

Partition of a set. A partition over a given set E is a set $P \subseteq \wp(E)$ s.t.:

- $\bigcup_{p_i \in P} p_i = E$
- $p_i \cap p_j = \emptyset$, for any $p_i, p_j \in P$ with $i \neq j$.
- For any $p \in P$, $p \neq \emptyset$

In other words, a partition covers E and is composed of disjoint subsets of E.

Equivalence relation. There is a bijection between the sets of partitions and the set of equivalence relations of a set. This 1-1-correspondence between P and R_P is given by $(e, e') \in R_P$ iff e and e' belongs to the same equivalence class of P [42, 72]. For example, given $P = \{\{1, 2, 3\}, \{4\}\}$, one has the relation $R_P = \{(1, 2), (1, 3), (2, 3), (1, 1), (2, 2), (3, 3), (4, 4)\}$ (omitting symmetry for the sake of readability).

The set of equivalence relations on any set T can be ordered by inclusion, if we consider a relation as a set of pairs of T, or also as the natural order on partitions, if we take the partition notation for the relations.

Ordering relation. A partition P_1 is finer than a partition P_2 (P_2 is coarser than P_1), written $P_1 \sqsubseteq P_2$ if any subset of P_1 is a subset of a subset in P_2. For example,

$$\{\{1, 3\}, \{2\}, \{4\}\} \sqsubseteq \{\{1, 2, 3\}, \{4\}\}$$

In fact, the sets of equivalence relations of a set T, or the set of partitions of T, is a lattice. The unit element (top) of this lattice denotes the fact that all objects are equivalent, i.e. all the attributes are in one class, while in the zero element (bottom) there are no two equivalent elements, i.e. each single element forms an equivalence class ($|T|$ classes of equivalence). Seen as sets of pairs $T \times T$, the top element contains precisely $T \times T$, whereas the zero element contains the sets $\{ (x, x) \mid \forall x \in T \}$.

We can define the meet of two equivalence relations, or two partitions, as follows:

Meet of two partitions. It is defined as the coarsest common refinement of two partitions. In other words, it is the intersection of the respective equivalence relations (omitting reflexivity for the sake of readability):

$$\begin{aligned} \{\{1,3\},\{2,4\}\} \sqcap \{\{1,2,3\},\{4\}\} &= \{\{1,3\},\{2\},\{4\}\} \\ \text{or } \{(1,3),(2,4)\} \cap \{(1,2),(1,3),(2,3)\} & \end{aligned}$$

The meet of two partitions is identical to the intersection of two equivalence relations seen as sets of pairs of tuples. Likewise, we can also define the join of two equivalence relations or partitions, which, again, can be seen as the union of two sets of pairs:

Join of two partitions. It is defined as the finest common coarsening of two partitions. In other words, it is the transitive closure of the union of the respective equivalence relations.

$$\begin{aligned} \{\{1,3\},\{2\},\{4\}\} \sqcup \{\{1,2\},\{3\},\{4\}\} &= \{\{1,2,3\},\{4\}\} \\ \text{or } \mathit{transitive\_closure}(\{(1,3)\} \cup \{(1,2)\}) &= \{(1,2),(1,3),(2,3)\} \end{aligned}$$

Finally, one should notice that the property $P_1 \sqcap P_2 = P_1 \Leftrightarrow P_1 \sqsubseteq P_2$ naturally holds (and the dual for join). Since the set of all partitions over a set forms a lattice $(D, \sqcap, \sqcup)$, it can be used as a description space of a pattern structure.

2.4.2 Partition Pattern Structure

Consider a tuple table T as a many-valued context (G, M, W, J) where $G = T$ corresponds to the set of objects ("rows"), $M = \mathcal{U}$ to the set of attributes ("columns"), $W = Dom$ the data domain ("all distinct values of the table") and $J \subseteq G \times M \times W$ a relation such that $(g, m, w) \in J$ also written $m(g) = w$ means that attribute m takes the value w for the object g [64]. In Table 2.3 (left), we have $d(t_4) = 3$.

We show how a partition pattern structure can be defined from a many-valued context (G, M, W, J) and show that its concept lattice is equivalent to the concept lattice of $\mathbb{K} = (\mathcal{B}_2(G), M, I)$ introduced above. Intuitively, formal objects of the pattern structure are the attributes of the many-valued context (G, M, W, J). Then, given an attribute $m \in M$, its description $\delta(m)$ is given by a partition over G such that any two elements g, h of the same class take the same values for the attribute m, i.e. $m(g) = m(h)$. The result is given in Figure 2.3 (middle). As such, descriptions obey the ordering of a partition lattice as described above. It follows that (G, M, W, J) can be represented as a pattern structure $(M, (D, \sqcap), \delta)$ where M is the set of original attributes, and $(D, \sqcap)$ is the set of partitions over G provided with the partition intersection operation $\sqcap$. An example of concept formation is given as follows, starting from set $\{a, d\} \subseteq M$:

$$\begin{aligned} \{a,d\}^{\square} &= \delta(a) \sqcap \delta(d) \\ &= \{\{t_1,t_3\},\{t_2,t_4\}\} \sqcap \{\{t_1,t_3\},\{t_2,t_4\}\} \\ &= \{\{t_1,t_3\},\{t_2,t_4\}\} \\ \{\{t_1,t_3\},\{t_2,t_4\}\}^{\square} &= \{m \in M | \{\{t_1,t_3\},\{t_2,t_4\} \sqsubseteq \delta(m)\} \\ &= \{a,d\} \end{aligned}$$

Hence, $(\{a, d\}, \{\{t_1, t_3\}, \{t_2, t_4\}\})$ is a pattern concept. The resulting pattern concept lattice is given in Figure 2.3 (right).

In the previous section, a many-valued context (G, M, W, J) was derived as the formal context $(\mathcal{B}_2(G), M, I)$ where $\mathcal{B}_2(G)$ represents any pair of objects, and $((g, h), m) \in I$ means that $m(g) = m(h)$. The resulting concept lattice is used to characterize the set of FDs [64]. A new result is that both structures $(\mathcal{B}_2(G), M, I)$ and $(M, (D, \sqcap), \delta)$ are equivalent, i.e. both collections of concepts are in 1-1-correspondence.

id	a	b	c	d
t_1	1	3	4	1
t_2	4	3	4	3
t_3	1	8	4	1
t_4	4	3	7	3

m	$\delta(m) \in (D, \sqcap)$
a	$\{\{t_1, t_3\}, \{t_2, t_4\}\}$
b	$\{\{t_1, t_2, t_4\}, \{t_3\}\}$
c	$\{\{t_1, t_2, t_3\}, \{t_4\}\}$
d	$\{\{t_1, t_3\}, \{t_2, t_4\}\}$

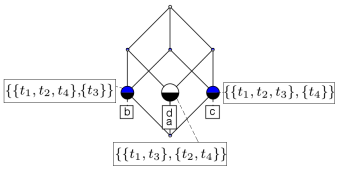


Figure 2.3: The original data (left), the resulting pattern structure (middle) and its pattern concept lattice (right)

Proposition 6 *(B, A) is a pattern concept of the partition pattern structure $(M, (D, \sqcap), \delta)$ if and only if (A, B) is a formal concept of the formal context $(\mathcal{B}_2(G), M, I)$ for all $B \subseteq M, A \subseteq \mathcal{B}_2(G)$ (equivalently A is a partition on G).*

Proof 4 *We first notice that a pattern $A \in D$ is a partition of the set of tuples, whereas elements of $\mathcal{B}_2(G)$ are sets of pairs of tuples. Yet, as we have seen in Subsection 2.3.1, objects in $\mathcal{B}_2(G)$ form an equivalence relation, and, therefore, partitions, which means that they correspond to patterns in D.*

Consider now that the concept lattices of the contexts $(\mathcal{B}_2(G), M, I)$ and $(M, \mathcal{B}_2(G), I)$ are equivalent, as they are built with "symmetric concepts": if (A, B) belongs to the first, (B, A) belongs to the second. With the context $(M, \mathcal{B}_2(G), I)$ and the pattern structure $(M, (D, \sqcap), \delta)$, the proposition holds since $B' = B^{\square}$ for all $B \subseteq M$:

$$\begin{aligned} B^{\square} &= \prod_{m \in B} \delta(m) \\ &= \bigcap_{m \in B} \{\, (t, t') \mid t(m) = t'(m) \,\} \quad \forall t, t' \in T \\ &= \{\, (t, t') \mid t(B) = t'(B) \,\} \quad \forall t, t' \in T \\ &= B' \end{aligned}$$

And symmetrically, $A' = A^{\square}$ for all $A \in D$, $A \subseteq \mathcal{B}_2(G)$:

$$\begin{aligned} A^{\square} &= \{m \in M \mid A \sqsubseteq \delta(m)\} \\ &= \{m \in M \mid \forall (t, t') \in A : (t, t') \in \delta(m)\} \quad \forall t, t' \in T \\ &= \{m \in M \mid \forall (t, t') \in A : t(m) = t'(m)\} \quad \forall t, t' \in T \\ &= A' \end{aligned}$$

Example. The pattern concept $(\{b\}, \{\{1, 2, 4\}, \{3\}\})$ is equivalent to the formal concept $(\{(1, 2), (1, 4), (2, 4)\}, \{b\})$. One should remark that pattern structures offer more concise intent representation when the set of tuples becomes very large, i.e. storing a partition instead of all pairs of objects that are together in a same class of the partition.

Moreover, there is an isomorphism between the concept lattice of (G, M, I) and the pattern concept lattice of $(G, (D, \sqcap), \delta)$. Then, the following proposition states that FDs can be characterized within the pattern concept lattice.

Proposition 7 *A functional dependency $X \to Y$ holds in a table T if and only if: $\{X\}^{\square} = \{XY\}^{\square}$ in the partition pattern structure $(M, (D, \sqcap), \delta)$.*

Proof 5 *First of all, we notice that $(t, t') \in X^{\square}$ if and only if $t(X) = t'(X)$, i.e. $\forall x \in X : t(x) = t'(x)$. We also notice that $\{X, Y\}^{\square} \subseteq \{X\}^{\square}$, as $\{X\} \subseteq \{X, Y\}$.*

(⇒) We prove that if $X \to Y$ holds in T, then, $\{X\}^{\square} = \{X,Y\}^{\square}$, i.e. $\{X\}^{\square} \subseteq \{X,Y\}^{\square}$. We take an arbitrary pair $(t,t') \in \{X\}^{\square}$, i.e. $t(X) = t'(X)$. Since $X \to Y$ holds, it implies that $t(XY) = t'(XY)$, and this implies that $(t,t') \in \{X,Y\}^{\square}$.

(⇐) We take an arbitrary pair $t,t' \in T$ such that $t(X) = t'(X)$. Therefore, we have that $(t,t') \in X^{\square}$, and by hypothesis, $(t,t') \in XY^{\square}$, i.e. $t(XY) = t'(XY)$. Since this is true for all pairs $t,t' \in T$ such that $t(X) = t'(X)$, it comes that $X \to Y$ holds in T.

Example. We consider a FD that holds in Table 2.1: $a \to d$. It is characterized from $(\mathcal{B}_2(G), M, I)$ as an attribute implication. It holds as well in $(M, \mathcal{B}_2(G), I)$ and $(M, (D, \sqcap), \delta)$ as an object implication.

Therefore, a naive algorithm that computes $\{\, X \to XY \mid \{X\}^{\square} = \{XY\}^{\square} \,\}$ for all $X, Y \subseteq \mathcal{U}$ would compute the Functional Dependencies that hold in a table. I can be noticed that we proposed a generic algorithm based on *PreviousClosure* [63] (currently under review).

2.5 Characterizing DMVDs with Pattern Structures

In order to show the flexibility of pattern structures to characterize dependencies, we now introduce how to handle a more general type of dependencies: degenerated multivalued dependencies (DMVDs). We have seen that the computation of functional dependencies is based on the equivalence relations (partitions) that are induced by an attribute. In order to compute DMVDs we consider now a *tolerance relation*. This relation is different from an equivalence relation in that it is symmetric and reflexive but not necessarily transitive. We define a tolerance relation on the set of tuples of a relation induced by an attribute:

Definition 23 *Let $a \in \mathcal{U}$ and let $\overline{a} = \mathcal{U} \setminus \{\, a \,\}$. The tolerance relation $\mathcal{R}_{\mathrm{T}}$ in a table T induced by a is:*

$$\mathcal{R}_{\mathrm{T}}(a) = \{(t_i, t_j) \in T \times T \mid i < j \text{ and } t_i(a) = t_j(a) \text{ or } t_i(\overline{a}) = t_j(\overline{a})\}$$

With the restriction $i < j$ we prevent pairs such as (t_i, t_i), or two symmetric pairs (t_i, t_j) and (t_j, t_i) from appearing in the representation of a relation, because, since symmetry and reflexivity hold, their presence is redundant. Note that the difference w.r.t. the definition of functional dependencies is the addition of the conjunctive clause $t_i(\overline{a}) = t_j(\overline{a})$.

Example: Consider the example of the table on the right. We see that $\mathcal{R}_{\mathrm{T}}(a) = \{(t_1, t_2), (t_2, t_3)\}$, (reflexivity and symmetry are omitted) but $(t_1, t_3) \notin \mathcal{R}_{\mathrm{T}}(a)$, which would hold because of transitivity.

id	a	b	c	d
t_1	1	1	1	1
t_2	1	2	2	2
t_3	3	2	2	2

Since tolerance relations are sets of pairs of tuples, if we define the meet and join between two tolerance relations as their set intersection and union, and order them by set inclusion, we have that the set of all possible tolerance relations is a complete lattice.

Given a tolerance relation, so called blocks of tolerance are defined as maximal sets of pairwise elements in correspondence (see [98] in FCA settings):

Definition 24 *Given a set G, a subset $K \subseteq G$, and a tolerance relation I on G, K is a* block of tolerance *if:*

(i) $\forall x, y \in K$ xIy (pairwise in correspondence)
(ii) $\forall z \notin K, \exists u \in K$ $\neg(zIu)$ (maximality)

For instance, the tolerance block $\{\, t_1, t_2, t_4 \,\}$ represents the set of pairs $\{\, (t_1, t_2), (t_1, t_4), (t_2, t_4) \,\}$. An attribute $m \in M$ is no longer described by a partition over the set of objects as in the previous section, but rather by a set of tolerance blocks. The pattern structure that we obtain, denoted by $(M, (D, \sqcap), \delta)$,

is such that $\delta(m)$ maps an attribute $m \in M$ to the set of tolerance blocks of the relation $\mathcal{R}_{\mathrm{T}}(m)$. The description space $(D, \sqcap)$ admits the same meet $\sqcap$ and the same ordering relation $\sqsubseteq$ as the partition lattice. Then, an example of concept formation is given as follows, starting from the set $\{a, b\} \subseteq M$:

$$\begin{aligned} \{a,b\}^{\square} &= \delta(a) \sqcap \delta(b) \\ &= \{\{t_1,t_3\},\{t_2,t_4\}\} \sqcap \{\{t_1,t_2,t_4\},\{t_1,t_3\}\} \\ &= \{\{t_1,t_3\},\{t_2,t_4\}\} \\ \{\{t_1,t_3\},\{t_2,t_4\}\}^{\square} &= \{m \in M \mid \{\{t_1,t_3\},\{t_2,t_4\}\} \sqsubseteq \delta(m)\} \\ &= \{a,b,c,d\} \end{aligned}$$

The resulting pattern structure along with its pattern concept lattice is given in Figure 2.4.

It can be noticed that, as for functional dependencies, a formal context can be built to characterize DMVDs, whose concept lattice is equivalent. The formal context can be written as $(M, \mathcal{B}_2(G), R)$ where $(m, (t_i, t_j)) \in R \iff (t_i, t_j) \in \mathcal{R}_{\mathrm{T}}(m)$. The resulting formal context of our example and its concept lattice are given in Figure 2.5. Here again, pattern structures offer more concise object descriptions with sets of blocks of tolerance instead of sets of pairs of tuples.

Now we can state how a DMVD $X \to Y$ holds in T according to the pattern structure $(M, (D, \sqcap), \delta)$.

Theorem 1 *Let $Z = \mathcal{U} \setminus X \setminus Y$. A DMVD $X \to Y$ holds in T if and only if*

$$\{X\}^{\square} = \{XY\}^{\square} \cup \{XZ\}^{\square}$$

Proof 6 *We assume that $(t, t') \in X^{\square}$ and $X' \subseteq X$ implies that $(t, t') \in X'^{\square}$. We also have that $Z = \mathcal{U} \setminus X \setminus Y$.*

($\Rightarrow$) We take two different tuples $t, t' \in X^{\square}$. We have two different options:

1. *$t(X) \neq t'(X)$. This implies, necessarily, that there is a subset of attributes $W \subseteq X$ such that $t(W) \neq t'(W)$, which implies that $t(\overline{W}) = t'(\overline{W})$. In this case, since $YZ \subseteq \overline{X}$ we have that for all $x \in YZ : (t, t') \in \delta(x)$, and, therefore, $(t, t') \in XYZ^{\square}$, i.e. $(t, t') \in \{XY\}^{\square} \cup \{XZ\}^{\square}$.*
2. *$t(X) = t'(X)$. Since $X \to Y$ holds in T, we have that $t(Y) = t'(Y)$ or $t(Z) = t'(Z)$. In the first case, we have that, for all $y \in Y : t(y) = t'(y)$, and then, $(t, t') \in \delta(y)$. This yields that $(t, t') \in XY^{\square}$. The case $t(Z) = t'(Z)$ is symmetric.*

In both cases we have that $(t, t') \in \{XY\}^{\square} \cup \{XZ\}^{\square}$.

$m \in M$	$\delta(m) \in (D, \sqcap)$
a	$\{\{t_1,t_3\},\{t_2,t_4\}\}$
b	$\{\{t_1,t_2,t_4\},\{t_1,t_3\}\}$
c	$\{\{t_1,t_2,t_3\},\{t_2,t_4\}\}$
d	$\{\{t_1,t_3\},\{t_2,t_4\})\}$

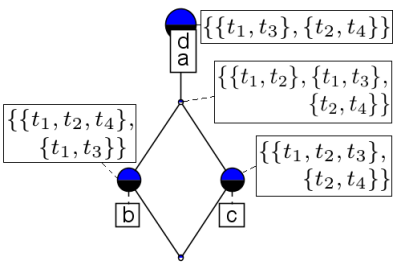


Figure 2.4: Characterizing DMVDs with a pattern concept lattice: The pattern structure (on the left) obtained by transforming Table 2.1 and its pattern concept lattice (on the right)

(⇐) We take two different tuples (t, t') such that $t(X) = t'(X)$. This implies that $(t, t') \in X^{\square}$, and, therefore, by hypothesis, that $(t, t') \in XY^{\square}$ or $(t, t') \in XZ^{\square}$. Both cases are symmetric, and we take the former one: $(t, t') \in XY^{\square}$, and in this case we have two different cases:

1. *There is a subset of attributes $W \in Y$ such that $t(W) \neq t'(W)$. In this case, we have that necessarily $t(\overline{W}) = t'(\overline{W})$. Since $Z \subseteq \overline{Y}$, then, $t(Z) = t'(Z)$ and the DMVD $X \to Y$ holds in T (by symmetry).*
2. *We have that $t(Y) = t'(Y)$, in which case the DMVD $X \to Y$ holds in T.*

We find here a method to check whether a DMVD holds in a table, similar to that described in Section 2.3. For instance, if we want to check if $a \to b$ we check if $a^{\square} = ab^{\square} \cup acd^{\square}$, which is true since $a^{\square} = \{(t_1, t_3), (t_2, t_4)\}$, $ab^{\square} = \{(t_1, t_3), (t_2, t_4)\}$ and $acd^{\square} = \{(t_1, t_3), (t_2, t_4)\}$. If we want to test whether $c \to a$, we see that $c^{\square} = \{(t_1, t_2), (t_1, t_3), (t_2, t_3), (t_2, t_4)\}$ whereas $ac^{\square} = \{(t_1, t_3), (t_2, t_4)\}$ and $bcd^{\square} = \{(t_1, t_3), (t_2, t_4)\}$ which implies that $c^{\square} \neq ac^{\square} \cup bcd^{\square}$, and by Theorem 1 means that $c \to a$ does not hold in T.

As we did in the previous section, we do not discuss how to enumerate all the DMVDs that hold in a table, although Theorem 1 states that a naive algorithm that computes $\{ X \to XY \mid \{X\}^{\square} = \{XY\}^{\square} \cup \{XZ\}^{\square} \}$ for all $X, Y \subseteq \mathcal{U}$ would be enough.

2.6 Conclusion

On one hand, the discovery of functional dependencies is an important topic in the field of databases. On the other, the discovery of implications is an attracting topic in formal concept analysis. We started our investigation from a known result that links both fields: functional dependencies can be characterized with formal concept analysis after a data transformation leading to a heavy data representation. Accordingly, we tackled the problem of avoiding such transformation by introducing partition pattern structures, a new conceptual structure that allows an equivalent characterization, but coming with better computational properties. Indeed, the empirical results show that, although the classical FCA approach performs well for small datasets, it is not scalable compared to partition pattern structures [20]. Since real-world datasets become larger and larger, this scalability is a more important feature than the speed concern for small datasets.

We continued our investigation with other kinds of dependencies, namely *approximate matching dependencies* and *order dependencies* and showed that it exists also a pattern structure where implications are in 1-1-correspondence [46, 18]. We have also proposed a generic algorithm and are currently experimenting it and comparing it with state of the art *ad hoc* algorithms. The results are encouraging for the moment.

$\mathbb{K}$	(1,2)	(1,3)	(1,4)	(2,3)	(2,4)	(3,4)
a		×			×	
b	×	×	×		×	
c	×	×		×	×	
d		×			×	

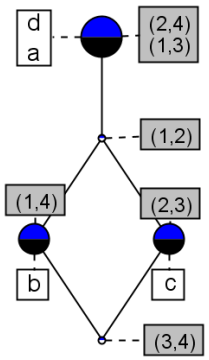


Figure 2.5: Characterizing DMVDs with FCA.

Chapter 3

Numerical Pattern Mining

This chapter introduces a novel type of numerical patterns, namely convex polygon patterns. Biclustering numerical is a particular way a defining numerical patterns in data, mainly used for applications in biology and recommender systems. In FCA and pattern mining, the kinds of patterns that are extracted are intervals: hyper-rectangles with sides parallel to the Euclidean plan axes, and the attributes are considered as independent. To allow more expressiveness, we investigate another type of patterns: polytopes.

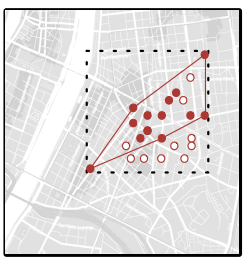
After recalling intervals patterns in Section 3.2, we formally introduce such patterns in Formal Concept Analysis (FCA) in Section 3.3. Then, we give all the basic bricks for mining convex polygons with exhaustive search and pattern sampling, and finally design several algorithms in Section 3.4 before to conclude. The experimental evaluation of the algorithms is not reported in this document but in the original article [22].

3.1 Introduction

Highly focused on over the past 20 years, pattern mining, the task of discovering interesting generalizations of object descriptions (subsets, subsequences, subgraphs, etc) has become a mature subfield in AI for knowledge discovery purposes [69]. When objects are given with a class label, discriminant patterns enable to elicit hypotheses from the data and to build intelligible classifiers [175].

Dealing with numerical data, and especially spatio-temporal data is still challenging. Algorithms supporting the correct, complete and non-redundant enumeration of particular shapes, say geometrical, have surprisingly not attracted much research interest [6]. Generally, numerical attributes (even spatial) are discretized, either in a pre-processing or on-the-fly during the execution (run) of a pattern enumeration algorithm (e.g. [73]), which has the consequence of considering geo-coordinates attributes independently and thus rectangular shape patterns occur. Still, such patterns were successfully used for mining numerical data (e.g. with most of the subgroup discovery approaches [53]), and spatial data (such as urban and mobility data see e.g. [23] and [92]).

The problem with rectangular shapes can be observed on the right hand side figure. Each object gives a POI (Point Of Interest) of a given type (Hotel, Restaurant, University, ...) and position. An interesting pattern is understood as a geographical area for which there is a sufficient number of points, high density, and a high proportion of objects of the same type. The candidate areas could have any shape. Rectangles, as being the products of intervals, have edges parallel to the plane axes: they may enclose both dense and sparse regions. Arbitrary polygons stick too much to the data and are hard to interpret. We consider convex polygons, a good trade-off for capturing high density areas.



Our contribution introduces a new type of patterns, convex polygons, with FCA tools [64], going

beyond the formalization of hyper-rectangles given by [98] in the early 1990s and introduced in pattern mining by [90]. We thus make precise the well-known notion of Galois connection as well as several polygon enumeration techniques and associated algorithms, using several concepts from computational geometry. We introduce several polygon constraints and experiment with our algorithms. We show that polygons give a better trade-off between area, density and homogeneity for mining spatial data. These findings give the basic bricks for any pattern mining algorithm dealing with, among others, a spatial attribute. The major problem with polygons is their worst-case exponential number (in number of input points). This is probably why they were not used in pattern mining until now: Exhaustive enumerations fail at considering large datasets even with 100 objects. We finally show that embedding our enumeration techniques in a recent pattern sampling technique (Monte Carlo Tree Search) enables us to discover high quality patterns very quickly in large datasets.

3.2 Interval Patterns

We recall first the formalization of interval patterns, or hyper-rectangles, in terms of FCA, for understanding next our formalization of convex polygon patterns.

Numerical dataset. A numerical dataset is given by a set of objects G, a set of numerical attributes $M = \{m_i\}_{1 \leq i \leq |M|}$, where the range of $m_i \in M$ is a finite set denoted by W_{m_i}, $m_i(g) = w$ means that w is the value of attribute m_i for object $g \in G$. Figure 3.1 plots 5 objects with 2 attributes on the Euclidean plane.

Interval pattern. An interval pattern is a box (hyper-rectangle) with sides *parallel to coordinate axes* formally defined as the Cartesian product of intervals $d = \langle [a_i, b_i] \rangle_{1 \leq i \leq |M|}$. An object g is in the image of an interval pattern d when $m_i(g) \in [a_i, b_i]$ $\forall i \in [\![1, |M|]\!]$. The support of d, denoted by $sup(d)$, is the set of objects in the image of d.

Interval pattern search space. The search space of interval patterns is the finite set D of all interval vectors $\langle [a_i, b_i] \rangle_{1 \leq i \leq |M|}$ with $a_i, b_i \in W_{m_i}$. The size of the search space is given by: $|D| = \prod_{i \in [\![1,|M|]\!]} (|W_{m_i}| \times (|W_{m_i}| + 1)/2)$.

Many patterns in D have exactly the same support: different hyper-rectangles contain the same objects. To avoid this redundancy, a closure operator can be defined, and only closed patterns which are unique for a given support are retained. This is achieved thanks to the formalism of pattern structures introduced by [62]. An *interval pattern structure* is given by $(G, (D, \sqcap), \delta)$ where G is the set of objects, $(D, \sqcap)$ the semi-lattice of object descriptions (boxes) and $\delta : G \rightarrow D$ a mapping that associates to each object $g \in G$, a vector of numerical intervals in the form of one-point interval $\delta(g) = \langle [m_i(g), m_i(g)] \rangle_{i \in [\![1,|M|]\!]}$. Elements of D are called *patterns* and are ordered as follows: $c \sqcap d = c \Leftrightarrow c \sqsubseteq d$. The *infimum* $\sqcap$ is defined as follows. Let $c = \langle [a_i, b_i] \rangle_{i \in [\![1,|M|]\!]}$ and $d = \langle [e_i, f_i] \rangle_{i \in [\![1,|M|]\!]}$ we have $c \sqcap d = \langle [min(a_i, e_i), max(b_i, f_i)] \rangle_{i \in [\![1,|M|]\!]}$ which is the minimal bounding box of the two boxes c and d. The two following operators $(.)^{\square}$, with $A \subseteq G$ and $d \in (D, \sqcap)$

$$d^{\square} = \{g \in G \mid d \sqsubseteq \delta(g)\} \quad A^{\square} = \bigsqcap_{g \in A} \delta(g)$$

form a Galois connection between $(2^G, \subseteq)$ and $(D, \sqsubseteq)$. $(.)^{\square\square}$ is a closure operator, i.e., monotone, extensive, idempotent. Closed patterns, i.e., such that $d = d^{\square\square}$ are smallest patterns/rectangles for a given support. We note that, for any description (box) $d \in D$, $d^{\square\square}$ represents the *minimal bounding box* of the support of d ($d^{\square}$, objects enclosed by d). Figure 3.1 shows an example where $G = \{g_1, ..., g_5\}$, we have $d = \langle [1, 2], [1, 4] \rangle$ (left figure) which support is $d^{\square} = \{g_1, g_2, g_3\}$. However d is not closed. Indeed, $d^{\square\square} = \{g_1, g_2, g_3\}^{\square} = \langle [1, 2], [1, 3] \rangle$ (right figure) which represents its closure (the minimal bounding box of $\{g_1, g_2, g_3\}$).

Interval patterns enumeration. Consider first data with a single attribute. To enumerate all interval patterns in $(D, \sqcap)$, [90] started from the top pattern, which is the minimal bounding box of all objects

Figure 3.1: Non-closed (left) and closed (right) interval pattern.

in G ($G^{\square}$). Then, as shown in Figure 3.2, at every step of the algorithm two minimal changes are applied (minimal left change (`minLeftChange`) and minimal right change (`minRightChange`)). To ensure a non-redundant generation, `minLeftChange` are not allowed after `minRightChange`. For $|M|$ numerical attributes, the algorithm is the same with two differences: (1) It considers a total order on the set of attributes M; (2) When a minimal change is applied to the attribute m_i, only attributes $m_j >= m_i$ can be refined in further steps from the generated pattern.

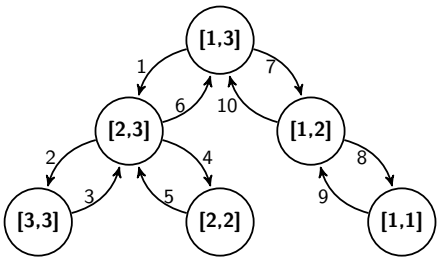


Figure 3.2: Depth-first traversal of $(D_{m_1}, \sqcap)$.

Closed interval patterns enumeration. To enumerate only closed patterns in $(D, \sqcap)$ (minimal bounding boxes), [90] adapted *CloseByOne* of [97]. Consider a pattern d generated by a change at attribute $m_j \in M$. Its closure is given by $d^{\square\square}$. If $d^{\square\square}$ differs from d for some attributes $m_i \in M$ such as $m_i < m_j$, then $d^{\square\square}$ has already been generated: the algorithm backtracks. Otherwise, it continues the enumeration from the closed pattern $d^{\square\square}$.

3.3 Convex Polygon Patterns

The choice of convex polygons instead of arbitrary ones is motivated by the fact that (i) it generalizes intervals (convex) and (ii) it is a natural way to avoid non-convex polygons which would over-fit the data.

3.3.1 Preleminaries

Convex polygon. A *convex polygon* P, represented as a sequence of points $[p_1, ..., p_h]$ in $\mathbb{R}^2$, is a simple polygon (not self-intersecting) where all interior angles are strictly less than π. The ordered point sequence $[p_1, ..., p_h]$ is denoted by $\overline{P}$ and is given in *counterclockwise order (ccw)*. Points p_i are called *extreme points* and *oriented line segments* $p_i p_{i+1}$ following this order are called *edges* of the polygon P (where $i + 1 = 1$ if $i = h$ and $i - 1 = h$ if $i = 1$).

Note that an oriented line segment AB subdivides $\mathbb{R}^2$ in 4 regions: (i) AB^+ (resp. (ii) AB^-) the open upper (resp. lower) half-plane of AB (*i.e.* $Q \in AB^+$ implies that ABQ is a triangle in *ccw* (resp. *cw*) order), (iii) AB^0 the set of points on the segment AB and (iv) the points that are collinear with A and B but outside AB.

Let $q \in \mathbb{R}^2$. An edge $p_i p_{i+1}$ is said to be *visible from* q (or q sees $p_i p_{i+1}$) iff $q \in p_i p_{i+1}^-$. A point q is in the enclosed area of the polygon P iff q does not see any edge of P ($q \in p_i p_{i+1}^+ \forall i \in \{1, ..., h\}$). Conventionally, $\emptyset$, points and segments are considered as convex polygons.

Convex hull. Given a finite set $E \subseteq \mathbb{R}^2$, the *convex hull* of E denoted by $ch(E)$ is the smallest convex set that contains E, which is a *convex polygon*. Note that $\overline{ch(E)} \subseteq E$. In other words, extreme points of the *convex hull* of E are in E. Application $ch : \mathbb{R}^2 \to \mathbb{R}^2$ is monotone, extensive, and idempotent, i.e., a closure operator.

Example. Figure 3.3 shows a polygon P where extreme points set $\overline{P} = [p_1, p_2, p_3, p_4, p_5, p_6]$. The point I is in the area enclosed by the polygon P. The point C_1 (resp. C_2) sees only the edge p_3p_4 (resp. p_4p_5). Generally spoken, the green triangle (resp. yellow triangle) represents all possible points that are only visible by the edge p_3p_4 (resp. p_4p_5). The point O sees the two edges p_3p_4 and p_4p_5.

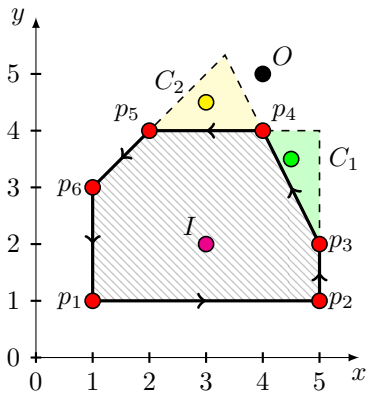


Figure 3.3: Convex polygon and edges visibilities.

3.3.2 Convex Polygon Pattern Structure

Spatial attribute. A spatial attribute m takes values in $\mathbb{R}^2$. Given a set of objects G, $m(g) = p$ means that $p \in \mathbb{R}^2$ is the value of attribute m for object $g \in G$. Analogically, for a subset $A \subseteq G$, $m(A) = \{m(g) \mid g \in A\} \subset \mathbb{R}^2$. For the sake of simplicity, we will consider now a dataset G having only *one spatial attribute* m. For ease of notations, g and $m(g)$ (G and $m(G)$) will be used interchangeably in what follows.

Convex polygon pattern. A convex polygon pattern d is a convex polygon $[p_i]_{1 \leq i \leq h}$ given in *ccw* order. An object g is in the image of a convex polygon pattern d when $m(g)$ is in the enclosed area formed by the polygon d. The support of d, denoted by $sup(d)$, is the set of objects $g \in G$ in the image of d. In Figure 3.3, $d = [p_1, p_2, ..., p_6]$ is a convex polygon pattern which support is $sup(d) = \{p_1, p_2, ..., p_6, I\}$.

Convex polygon pattern structure. It is defined as follows: $(G, (D, \sqcap), \delta)$ where G is the set of objects described by one spatial attribute m, $(D, \sqcap)$ is the semi-lattice of object descriptions (convex polygons) where $D = \{\overline{ch(m(A))} | A \subseteq G\}$. The mapping $\delta : G \to D$ takes each object $g \in G$ to $\delta(g) = [m(g)]$ (a degenerate polygon with a singleton point). The *infimum* $\sqcap$ is defined as follows. Let c and d be two descriptions in D, we have $c \sqcap d = \overline{ch(c \cup d)}$. Elements of $(D, \sqcap)$ are ordered as follows: $c \sqsubseteq d \Leftrightarrow c \sqcap d = c \Leftrightarrow d \subseteq ch(c)$. Thus, the Galois operators $(.)^\square$ are defined as follows, with $A \subseteq G$ and $d \in (D, \sqcap)$: $d^\square = \{g \in G \mid m(g) \in ch(d)\}$ and $A^\square = \overline{ch(m(A))}$. A pair (A, d) s.t. $A^\square = d$ and $d^\square = A$ is a concept. Both A and d are closed under $(.)^{\square\square}$. The set A contains all objects, while d contains only extreme points in *ccw* order.

Mining convex polygons with constraints. There are 2^n convex polygons in the worst case (n points on a circle). Not all of them are interesting, we thus define several constraints a pattern shall respect. The problem is then, given a set of points in $\mathbb{R}^2$, to find all polygon pattern concepts (A, d)

respecting one or more of the following constraints, such as shape complexity ($|d| \leq \delta_{compl}$), minimal support ($|A| \geq \delta_{supp}$), minimal/maximal perimeter ($perim(d) \geq$ or $\leq \delta_{perim}$), and minimal/maximal area ($area(d) \geq$ or $\leq \delta_{area}$). We also wish patterns that maximize a class homogeneity (e.g. low Gini) and density ($|A|/area(d)$).

By using minimal support we avoid considering polygons with too few points. The density has to be understood as a relative support (support normalized by area). The number of extreme points characterizes the complexity of the polygon in terms of interpretation (point, segment, triangle, quadrilateral, ...). The simpler the form the better by principle of parsimony: Minimizing the shape complexity may also avoid over-fitting the data when searching for discriminant patterns. Controlling perimeter and area is also important: It allows one to express different types of patterns (e.g. avoiding or forcing thin polygons, with a large perimeter and small surface).

3.4 Algorithms

Let G be a finite subset of $\mathbb{R}^2$. We propose three algorithms for mining convex polygon patterns. The first one enumerates object subsets in a bottom-up way (from $\emptyset$ to G) with closures. The next one considers a top-down enumeration and does not compute costly closures. The last one considers a bottom-up enumeration of polygons by shape complexity (points, then segments, then triangles...) and performs the best.

3.4.1 Enumerating Point Sets

As the operators $((.)^{\square}, (.)^{\square})$ form a Galois connection, the set of all pattern concepts is given by $\mathcal{C} = \{(A^{\square\square}, A^{\square}), \forall A \subseteq G\}$. The $A^{\square}$ are precisely the polygon patterns and can thus be obtained by enumerating closed object subsets with the generic algorithm *CloseByOne (CbO)* of [97] and its modern efficient software realization [12]. The principle is the following. The lattice $(2^G, \subseteq)$ is explored with a DFS starting from $\emptyset$. A total order on G is provided, e.g. $g_1 < g_2 < ... < g_n$. A new object set A is generated from a previous one by adding next object g w.r.t. $<$ and $(A^{\square\square}, A^{\square})$ is the resulting pattern concept. The concept is discarded (DFS backtrack) if an object smaller than g w.r.t. $<$ is added. It is proven that this algorithm outputs the correct, complete and non-redundant collection of closed patterns. Namely, ExtCbO (Algorithm 2) is the version of CbO where concepts are computed w.r.t. increasing generality order, i.e., by adding one object at a time. Algorithm ExtCbO will serve as a baseline.

Algorithm 2 ExtCbO

```
1: Let (G, ≤) be a totally-ordered finite subset of ℝ²
2: procedure ExtCbO( )
3:     traverse(∅,∅,1)
4: end procedure
5: procedure traverse(c,d,pos)                    ▷ c is the extent, d the intent
6:     Print(c,d)                                 ▷ visit pattern concept (c, d)
7:     for i ∈ pos, ..., |G| and g_i ∉ c do
8:         c_new ← c ∪ {g_i}
9:         d_new ← c_new^□                        ▷ Compute convex hull
10:        c_new ← d_new^□
11:        if ∀j ∈ [[1, i]] : g_j ∉ c → g_j ∉ c_new then
12:            traverse(c_new,d_new,i+1)
13:        end if
14:    end for
15: end procedure
```

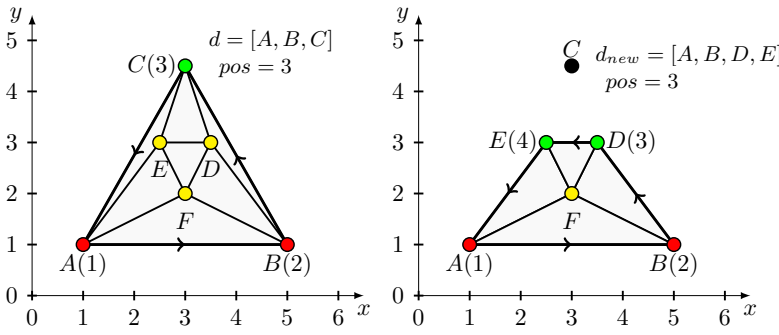


Figure 3.4: DelaunayEnum extreme point deletion.

3.4.2 Updating a Delaunay Triangulation

The enumeration starts from the most general pattern $G^{\square}$. Minimal changes are applied to obtain the next pattern to ensure the correctness and completeness of the enumeration: all convex polygons and only them are visited. Let us consider the convex polygon pattern $d = [p_1, ..., p_i, ..., p_{|d|}]$. A minimal change to d consists in removing an extreme point (as done with interval patterns). There are $|d|$ of such minimal changes for a polygon d for obtaining next smaller polygons $e = d \backslash \{d[i]\}$ s.t. $d \sqsubset e$, $i \in [\![1, |d|]\!]$. When a new pattern e is generated, we need to compute its convex hull to discover its extreme points $d_{new} = e^{\square\square}$ and continue the enumeration.

This algorithm is again an instance of *CloseByOne*, but follows a top-down enumeration. Sadly, it still forces to compute the convex hull $e^{\square}$ at each step. Fortunately, it can be avoided with a *Delaunay triangulation*.

Delaunay triangulation. The *Delaunay triangulation* of a set of points P denoted by $DT(P)$ is a partition of the *convex hull* $ch(P)$ into $\mathcal{O}(n)$ triangles such that: *(i)* Triangles vertices are in P, *(ii)* No point in P is inside the circumcircle of any triangle in $DT(P)$ besides the vertices of the triangle. The complexity of computing $DT(P)$ is $\mathcal{O}(|P| \cdot log(|P|))$ [172].

Algorithm DelaunayEnum**.** The key idea is the following. Efficiently computing $e^{\square} = (d \backslash \{d[i]\})^{\square}$, that is the next closed pattern obtained from removing the extreme point i of d, is equivalent to computing $DT(d \setminus \{d[i]\})$ from $DT(d)$. During this computation, one can easily update the sequence of extreme points. [51] answered this problem with an efficient algorithm ($\mathcal{O}(k \cdot log(k))$ complexity), where k is the number of points sharing an edge with p (neighbor points).

Our algorithm DelaunayEnum (Algorithm 3) enumerates *Delaunay triangulations dt* and maintains at each step the sequence of extreme points d. It starts from pattern $G^{\square} = \overline{ch(G)}$ (line 4). Consider a step in the enumeration where d is the current pattern, dt the current triangulation and c the current image of d. We remove successively extreme points $p \in d$ and we update the Delaunay triangulation dt_{new} based on dt (line 11), the new description d_{new} (sequence of extreme points) (line 12) and the new support c_{new} by removing objects with value p from c (line 13). To avoid redundancy (avoid visiting the same pattern twice), removal of extreme points before the last removed extreme point p are not allowed (argument *pos* in Algorithm 3) in further steps. In order to do that simply, when the extreme point sequence d is updated upon extreme point i removal, we only insert the new extreme points (extreme points not in d) at the position i while keeping the same order (*counterclockwise*). Figure 3.4 (from left to right) shows an example of how the algorithm updates the description $d = [A, B, C]$ upon removal of the extreme point C (position 3) and produces the description $d_{new} = [A, B, \underline{D}, \underline{E}]$ where $\underline{D}$ and $\underline{E}$ denote the new extreme points that replaced C. The enumeration continues by removing D.

Algorithm 3 DelaunayEnum

```
1: Let G be a finite subset of ℝ²
2: procedure DelaunayEnum( )
3:     dt ← Delaunay(G)
4:     d ← dt̄                                   ▷ dt̄ = G^□ = ch(G)
5:     traverse(G,d,dt,1)
6: end procedure
7: procedure traverse(c,d,dt,pos)
8:     Print(c,d)                               ▷ visit pattern concept (c, d)
9:     for i ∈ pos, ..., |d| do
10:        p ← d[i]
11:        dt_new ← Delaunay_remove(dt,p)
12:        d_new ← dt̄_new                       ▷ d_new[1] = d[1] or d_new[|d_new|] = d[|d|]
13:        c_new ← c\[p]^□                      ▷ [p]^□ = {g ∈ G | m(g) = p}
14:        traverse(c_new,d_new, dt_new,i)
15:    end for
16: end procedure
```

3.4.3 Enumerating Simpler Shapes First

Until now, elements of $(D, \sqsubseteq)$ were enumerated w.r.t. polygon inclusion order. Now we consider the *poset* $(D, \bar{\sqsubseteq})$, where polygons are ordered w.r.t. extreme points inclusion: $c \,\bar{\sqsubseteq}\, d \Leftrightarrow c \supseteq d$. Intuitively, the i^{th} level of $(D, \bar{\sqsubseteq})$ contains all polygons with i extreme points ($i = 3$ are the triangles, $i = 4$ the convex quadrilaterals, ...). Enumerating simpler shapes first is interesting as they are easier to interpret and less stick to the data points (prevent overfitting). Note that the new order $\bar{\sqsubseteq}$ will change only the enumeration order.

In order to enumerate all the polygons w.r.t. the new order $\bar{\sqsubseteq}$ in a *bottom-up* fashion as simpler shapes are preferred, one can simply adapt ExtCbO. Indeed, polygons are represented as extreme points sets which can be enumerated starting from $\emptyset$. However, it requires to compute at each step the convex hull of the set of points (closure) which is extremely slow. To avoid this, we propose an enumeration technique relying on basic geometry: points used to extend a pattern produce a *new closed pattern* for sure. That is to say: (i) there is no need to compute the convex hull and (ii) there is no need to discard a pattern as done in ExtCbO since every generated pattern is new. This can be done thanks to *pattern candidate maintenance* as explained below.

In what follows, without loss of generality and for the sake of simplicity, elements of G are pairwise distinct. Moreover, for any oriented line segment AB: AB^+, AB^- and AB^0 will denote finite sets where only points of G are considered (rather than $\mathbb{R}^2$).

Pattern candidate points array. Consider a convex pattern $d = [p_1, ..., p_h]$ where $\forall i \in [\![1, h]\!] : p_i \in G$. We define a same-size array n_d called *candidate points array* where $n_d[i]$ gives the set of points that are *visible from and only from* the edge $p_i p_{i+1}$. Formally, if $h \geq 2$:

$$n_d[i] = p_i p_{i+1}^- \cap (\bigcap_{j \neq i} p_j p_{j+1}^+)$$

In case of $h = 1$, we have $n_d[1] = G \backslash \{p_1\}$. The proposition below gives a simpler formula for $n[i]$ when $h \geq 2$:

Proposition 8 *If $h \geq 2$, $\forall i \in [\![1, h]\!]$ we have:*

$$n_d[i] = p_i p_{i+1}^- \cap p_{i-1} p_i^+ \cap p_{i+1} p_{i+2}^+$$

Extending a pattern and candidate maintenance. We have the following proposition:

Proposition 9 $\forall i \in [\![1, h]\!]$, $\forall q \in n_d[i]$, *we have:*

$$d \sqcap [q] = [p_1, p_i, q, p_{i+1}..., p_h]$$

In other words, extending a pattern d by adding a candidate point q creates a new pattern $e = d \sqcap [q]$ s.t. $e \sqsubseteq d$.

Proposition 10 gives a method to compute the support and the candidate points of the new pattern e:

Proposition 10 *If $h \geq 2$, $\forall i \in [\![1, h]\!]$, $\forall q \in n_d[i]$, if $e = d \sqcap [q]$ and n_e its candidate points array, we have:*

$$\begin{aligned}
e^{\square} &= d^{\square} \cup [p_i, q, p_{i+1}]^{\square} \\
n_e[i] &= n_d[i] \cap p_i q^- \cap q p_{i+1}^+ \\
n_e[i+1] &= n_d[i] \cap q p_{i+1}^- \cap p_i q^+ \\
n_e[i-1] &= n_d[i-1] \cap p_i q^+ \\
n_e[i+2] &= n_d[i+1] \cap q p_{i+1}^+ \\
n_e[i+k] &= n_d[i+k-1] \text{ if } 3 \leq k \leq h-i+1 \\
n_e[i-k] &= n_d[i-k] \text{ if } 2 \leq k < i
\end{aligned}$$

Note that $[p_i, q, p_{i+1}]^{\square}$ is the subset of objects enclosed in the area formed by the triangle $[p_i, q, p_{i+1}]$. Formally: $[p_i, q, p_{i+1}]^{\square} = p_i q^0 \cup q p_{i+1}^0 \cup p_i p_{i+1}^0 \cup (p_i p_{i+1}^- \cap p_i q^+ \cap q p_{i+1}^+)$.

Propositions 8 and 9 are direct results from planar geometry [129], while Proposition 10 is a direct reformulation of Proposition 8.

Figure 3.3 shows a description $d = [p_1, p_2, p_3, p_4, p_5, p_6]$ with candidate points $n_d = [\emptyset, \emptyset, \{C_1\}, \{C_2\}, \emptyset, \emptyset]$. More generally, every point that falls in the green (resp. yellow) zone is a candidate point for the edge p_3p_4 (resp. p_4p_5). However, the point O is not a candidate point for any edge. Indeed, O sees two edges p_3p_4 and p_4p_5.

Algorithm ExtremePointsEnum**.** The algorithm ExtremePointsEnum (Algorithm 4) follows almost the same enumeration principle as that of ExtCbO. The difference lies in maintaining the list of object candidates that can be added to a pattern d to generate patten $e \sqsubset d$. The other difference is that the two first levels are enumerated in BFS manner. This allows one to build the segment index S that for each distinct object pair $g_i g_j$ of G stores the support $g_i g_j^0$ (objects in segment $g_i g_j$) and its candidates $g_i g_j^-$ and $g_i g_j^+$. Note that the segment index S can be seen as a *strictly upper triangular matrix*. The algorithm continues in a *DFS-fashion* to enumerate higher levels ($k \geq 2$). To prevent redundancy, an arbitrary total order on G is provided, e.g. $g_1 < g_2 < ... < g_n$. When extending a pattern $d = [p_1, ..., p_h]$, which candidate points array is n_d, with a point $q \in n_d[i]$, only points q s.t. $p_j < q$ $\forall j \in [\![1, h]\!]$ are considered.

Detailed example. Figure 3.5 gives a detailed step-by-step partial enumeration of convex patterns related to G w.r.t. $\sqsubseteq$ partial order. In each subfigure, red points are *extreme points*, yellow points are part of the support of the pattern but not extreme points, green points are candidate points and black points are not candidates (points located in the red zones).

3.5 Conclusion

In pattern mining, hyper-rectangles cannot always properly capture interesting areas although they are widely used. We formally defined convex polygon patterns by means of *Formal Concept Analysis (FCA)* and proposed three enumeration techniques. We do not report experimental results of our algorithms in this document. It turns out that they do now allow to consider large dataset. This is not the weakness

Algorithm 4 ExtremePointsEnum

Let $(G, \leq)$ be a totally-ordered set of pairwise distinct points
Let S be the segment index
procedure traverse(c, d, n, pos)
    Print(c,d) ▷ visit pattern concept (c, d)
    for $i \in 1, ..., |d|$ **do**
        for $j \in n[i]$ **and** $j >= pos$ **do** ▷ $j \geq pos$ for non redundancy
            $s1 \leftarrow S[d[i]][j]$ ▷ first new segment $d[i] \rightarrow j$
            $s2 \leftarrow S[j][d[i+1]]$ ▷ second new segment $j \rightarrow d[i+1]$
            $d_{new} \leftarrow [d[1], ..., d[i], j, d[i+1], ..., d[|d|]]$
            $c_{new} \leftarrow c \cup s1^0 \cup s2^0 \cup (n[i] \cap s1^+ \cap s2^+)$
            $n_{new} \leftarrow [n[1], ..., n[i], n[i], n[i+1], ..., n[|d|]]$
            $n_{new}[i] \leftarrow n[i] \cap s1^- \cap s2^+$
            $n_{new}[i+1] \leftarrow n[i] \cap s2^- \cap s1^+$
            $n_{new}[i-1] \leftarrow n[i-1] \cap s1^+$
            $n_{new}[i+2] \leftarrow n[i+1] \cap s2^+$
            traverse(c_{new},d_{new}, n_{new},$j+1$)
        end for
    end for
end procedure
procedure ExtremePointsEnum( )
    Print($\emptyset$,$\emptyset$) ▷ visit the empty pattern concept $(\emptyset, \emptyset)$
    Enumerate in *BFS-fashion* all distinct points G
    Compute segment index S
    for $i \in 1, ..., |P| - 1$ **do**
        for $j \in i+1, .., |P|$ **do**
            $s \leftarrow S[i, j]$
            traverse($s^{\square}$,$[i, j]$, $[s^-, s^+]$,j+1)
        end for
    end for
end procedure

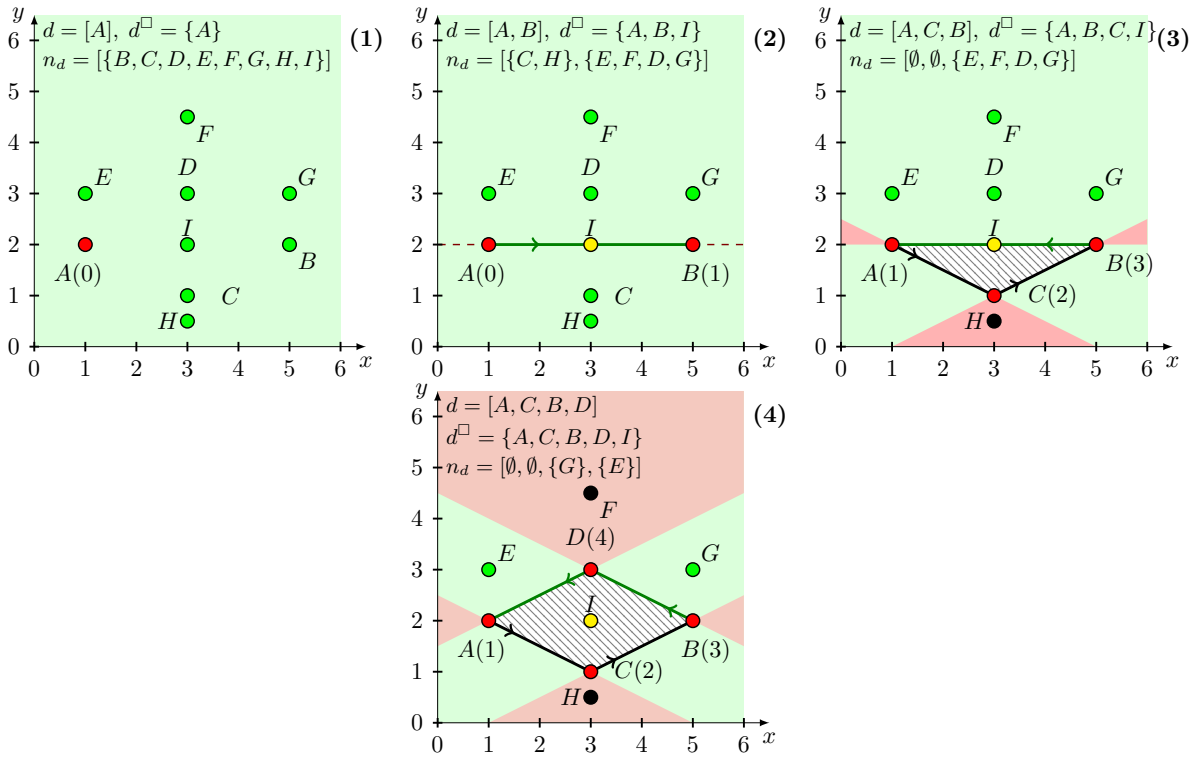


Figure 3.5: ExtremePointsEnum algorithm step-by-step enumeration.

of the algorithm, but the size of the search space. However, these enumeration techniques can be used in sampling algorithm, e.g. during a random walk. We successfully experimented the Monte Carlo Tree Search of convex polygons using the enumeration of simpler shapes first: We mined social network points of interest with the aim to discover regions that discriminate a class label (types of points of interest). Such approach is actually detailed in the next part for pattern mining in general, and interval patterns and itemsets in particular.

Part II

Pattern Mining and Subgroup Discovery

Chapter 4

Pattern Discovery with Monte Carlo Tree Search

This chapter introduces Subgroup Discovery: the task of finding patterns that describe well a class label and not others (Section 4.1). Section 4.2 then formalizes the issues we address: (i) scalability and (ii) diversity/redundancy: it is impossible in many cases to perform an exhaustive search of the pattern space, while heuristic methods have difficulties to sample this search space (diversity) and can return many variations of the same pattern (redundancy). Section 4.3 then smoothly presents MCTS, a recent exploration technique, while Section 4.4 deeply details its adaptation for pattern mining. We omit here our experimental evaluation which can be found in the original article [30] but discuss the advantages and disadvantages over other existing methods and our perspectives (Section 4.5).

4.1 Introduction

The discovery of patterns, or descriptions, which discriminate a group of objects given a target (class label) has been widely studied as overviewed by [127]. Discovering such descriptive rules can be formalized as the so-called subgroup discovery task (SD introduced by [167]). Given a set of objects, each being associated to a description and a class label, a subgroup is a description generalization whose discriminating ability is evaluated by a quality measure (F1-score, accuracy, etc). In the last two decades, different aspects of SD have been studied: The description and target languages (quantitative, qualitative, etc.), the algorithms that enable the discovery of the best subgroups, and the definition of measures that express pattern interestingness. These directions of work are closely related and many of the pioneer approaches were *ad hoc* solutions lacking from easy implementable generalizations (see for examples the surveys of [127] and [53]). SD still faces two important challenges: First, how to characterize the interest of a pattern? Secondly, how to design an accurate heuristic search technique when exhaustive enumeration of the pattern space is unfeasible?

[105] introduced a more general framework than SD called exceptional model mining (EMM). It tackles the first issue. EMM aims to find patterns that cover tuples that locally induce a model that substantially differs from the model of the whole dataset, this difference being measured with a quality measure. This rich framework extends the classical SD settings to multi-labeled data and it leads to a large class of models, quality measures, and applications [159, 53, 92]. In a similar fashion to other pattern mining approaches, SD and EMM have to perform a heuristic search when exhaustive search fails. The most widely used techniques are *beam search* [159, 116], *genetic algorithms* [48, 110], and *pattern sampling* [122, 23].

The main goal of these heuristics is to drive the search towards the most interesting parts, i.e., the regions of the search space where patterns maximize a given quality measure. However, it often happens

that the best patterns are *redundant*: They tend to represent the same description, almost the same set of objects, and consequently slightly differ on their pattern quality measures. Several solutions have been proposed to filter out redundant subgroups, e.g. as did [35, 159, 116, 31]. Basically, a neighboring function enables to keep only local optima. However, one may end up with a pattern set of small cardinality: This is the problem of *diversity*, that is, many local optima have been missed.

Let us illustrate this problem on Figure 4.1. The search space of patterns, a lattice, hides several local optima (patterns maximizing a pattern quality measure in a neighborhood). Figure 4.1(a) presents such optima with red dots, surrounded with redundant patterns in their neighborhood. Given the minimal number of objects a pattern must cover, exhaustive search algorithms, such as SD-Map [15, 14], are able to traverse this search space efficiently: The monotonocity of the minimum support and upper bounds on some quality measures such as the *weighted relative accuracy* ($WRAcc$) enable efficient and safe pruning of the search space. However, when the search space of patterns becomes tremendously large, either the number of patterns explodes or the search is intractable. Figure 4.1(b) presents beam-search, probably the most popular technique within the SD and EMM recent literature. It operates a top-down level-wise greedy exploration of the patterns with a controlled level width that penalizes diversity (although several enhancements to favor diversity have been devised [159, 160, 116]). Genetic algorithms have been proposed as well [140, 130, 40]. They give however no guarantees that all local optima will be found and they have been designed for specific pattern languages and quality measures [110]. Finally, pattern sampling is attractive as it enables direct interactions with the user for using his/her preferences to drive the search [28, 122]. Besides, with sampling methods, a result is available anytime. However, traditional sampling methods used for pattern mining need a given probability distribution over the pattern space which depends on both the data and the measure and may be costly to compute [28, 122]. Each iteration is independent and draws a pattern given this probability distribution (Figure 4.1(c)).

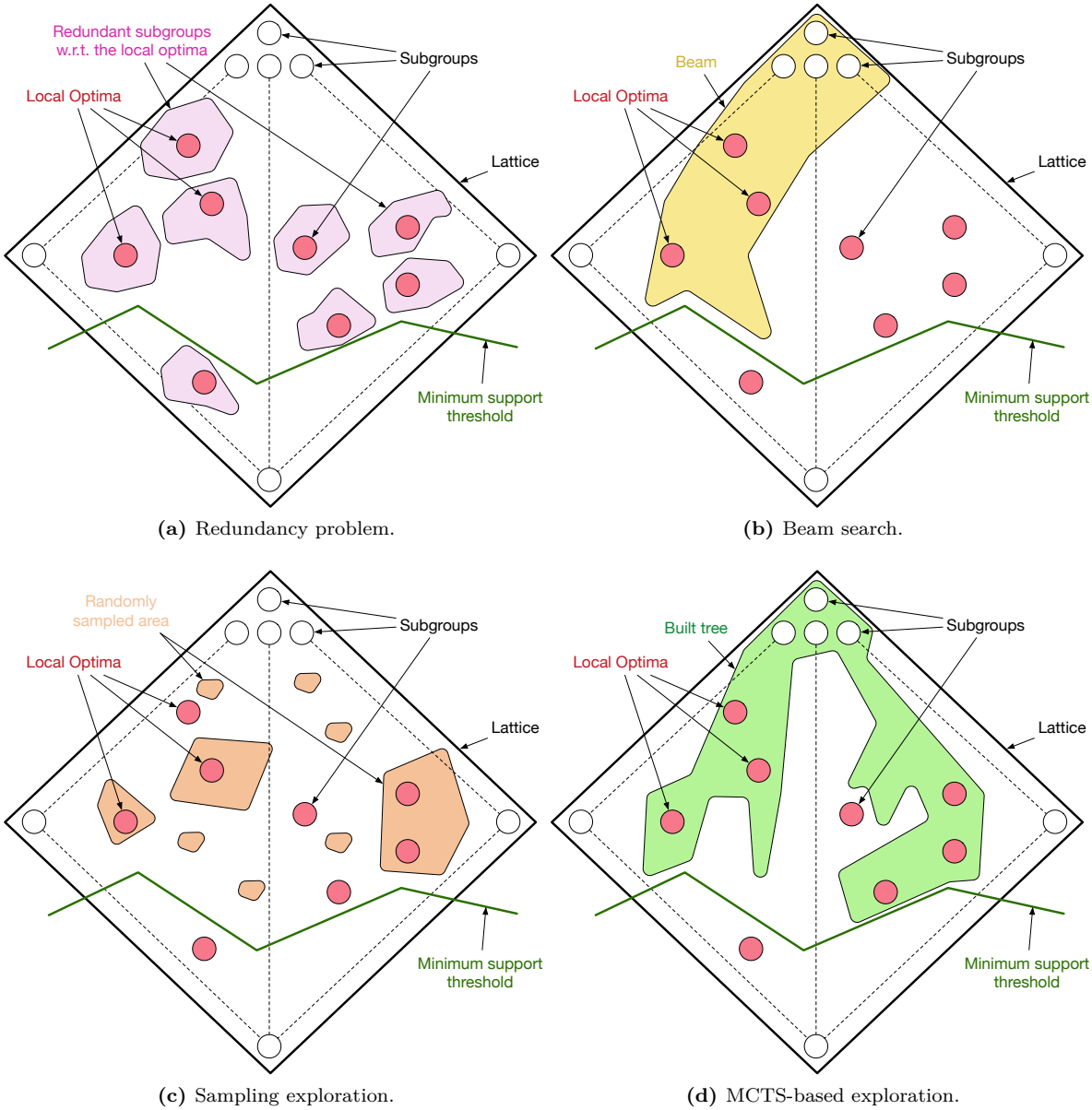
We propose to support subgroup discovery with a novel search method, Monte Carlo tree search (MCTS). It has been mainly used in AI for domains such as games and planning problems, that can be represented as trees of sequential decisions [36]. It has been popularized as definitively successful for the game of Go in [145]. MCTS explores a search space by building a game tree in an incremental and asymmetric manner: The tree construction is driven by random simulations and an exploration/exploitation trade-off provided by the so called upper confidence bounds (UCB) [95]. The construction can be stopped anytime, e.g., when a maximal budget is reached. As illustrated on Figure 4.1(d), our intuition for pattern mining is that MCTS searches for some local optima, and once found, the search can be redirected towards other local optima. This principle enables *per se* a diversity of the result set: Several high quality patterns covering different parts of the data set can be extracted. More importantly, the power of random search leads to *anytime mining*: A solution is always available, it improves with time and it converges to the optimal one if given enough time and memory budget. This is a *best-first search*. Given a reasonable time and memory budget, MCTS quickly drives the search towards a diverse pattern set of high quality. Interestingly, it can consider, in theory, any pattern quality measure and pattern language (in contrast to current sampling techniques as developped by [28, 122]).

Our main contribution is to introduce MCTS for subgroup discovery and pattern mining in general. Revisiting MCTS in such a setting is not simple and it requires to define smart new policies. We show through an extensive set of experiments that MCTS is a viable solution for a pattern mining task and that it outperforms the state-of-the-art approaches (exhaustive search, beam search, genetic algorithm, pattern sampling) when dealing with large search space of numerical and nominal attributes and for different quality measures.

4.2 Pattern Set Discovery

There exists several formal pattern mining frameworks and we choose here subgroup discovery to illustrate our purpose. We provide some basic definitions and then formally define pattern set discovery.

Definition 25 (Dataset $\mathcal{D}(\mathcal{O}, \mathcal{A}, \mathcal{C}, class)$) *Let $\mathcal{O}$, $\mathcal{A}$ and $\mathcal{C}$ be respectively a set of objects, a set of attributes, and a set of class labels. The domain of an attribute $a \in \mathcal{A}$ is $Dom(a)$ where a is either*



(a) Redundancy problem.

(b) Beam search.

(c) Sampling exploration.

(d) MCTS-based exploration.

Figure 4.1: Illustration of different SD search algorithms.

nominal or numerical. The mapping class $: \mathcal{O} \mapsto \mathcal{C}$ *associates each object to a unique class label.*

A subgroup can be represented either by a description (the pattern) or by its coverage, also called its extent.

Definition 26 (Subgroup) *The description of a subgroup (a pattern), is given by* $d = \langle f_1, \ldots, f_{|\mathcal{A}|} \rangle$ *where each* f_i *is a restriction on the value domain of the attribute* $a_i \in \mathcal{A}$*. A restriction for a nominal attribute* a_i *is a symbol* $a_i = v$ *with* $v \in Dom(a_i)$*. A restriction for a numerical*[4] *attribute* a_i *is an interval* $[l, r]$ *with* $l, r \in Dom(a_i)$*. The description* d *covers a set of objects called the extent of the subgroup, denoted* $ext(d) \subseteq \mathcal{O}$*. The support of a subgroup is the cardinality of its extent:* $supp(d) = |ext(d)|$*.*

[4]We consider the finite set of all intervals from the data, without greedy discretization. As shown later, better patterns can be found in that case, when using only MCTS on large datasets.

Table 4.1: Toy dataset

ID	a	b	c	$class(.)$
1	150	21	11	l_1
2	128	29	9	l_2
3	136	24	10	l_2
4	152	23	11	l_3
5	151	27	12	l_2
6	142	27	10	l_1

The subgroup search space is structured as a lattice.

Definition 27 (Subgroup search space) *The set of all subgroups forms a lattice, denoted as the poset $(\mathcal{S}, \preceq)$. The top is the most general pattern, without restriction. Given any $s_1, s_2 \in \mathcal{S}$, we note $s_1 \prec s_2$ to denote that s_1 is strictly more specific, i.e. it contains more stringent restrictions.*

If follows that $ext(s_1) \subseteq ext(s_2)$ when $s_1 \preceq s_2$.

The ability of a subgroup to discriminate a class label is evaluated by means of a quality measure. The weighted relative accuracy (WRAcc), intoduced by [102], is among the most popular measures for rule learning and subgroup discovery. Basically, WRAcc considers the precision of the subgroup w.r.t. to a class label relatively to the appearance probability of the label in the whole dataset. This difference is weighted with the support of the subgroup to avoid to consider small ones as interesting.

Definition 28 (WRAcc) *Given a dataset $\mathcal{D}(\mathcal{O}, \mathcal{A}, \mathcal{C}, class)$, the WRAcc of a subgroup d for a label $l \in Dom(\mathcal{C})$ is given by:*

$$WRAcc(d, l) = \frac{supp(d)}{|\mathcal{O}|} \times \left(p_d^l - p^l\right)$$

where $p_d^l = \frac{|\{o \in ext(d) | class(o) = l\}|}{supp(d)}$ and $p^l = \frac{|\{o \in \mathcal{O} | class(o) = l\}|}{|\mathcal{O}|}$.

WRAcc returns values in $[-0.25, 0, 25]$, the higher and positive, the better the pattern discriminates the class label. Many quality measures other than WRAcc have been introduced in the literature of rule learning and subgroup discovery (Gini index, entropy, F score, Jaccard coefficient, etc. [[3]]). Exceptional model mining (EMM) considers multiple labels (label distribution difference in [159], Bayesian model difference in [54], etc.). The choice of a pattern quality measure, denoted φ in what follows, is generally application dependant as explained by [60].

Example 14 *Consider the dataset in Table 4.1 with objects in $\mathcal{O} = \{1, ..., 6\}$ and attributes in $\mathcal{A} = \{a, b, c\}$. Each object is labeled with a class label from $\mathcal{C} = \{l_1, l_2, l_3\}$. Consider an arbitrary subgroup with description $d = \langle [128 \leq a \leq 151], [23 \leq b \leq 29] \rangle$. Note that, for readability, we omit restrictions satisfied by all objects, e.g., $[9 \leq c \leq 12]$, and thus we denote that $ext(\langle\rangle) = \mathcal{O}$. The extent of d is composed of the objects in $ext(d) = \{2, 3, 5, 6\}$ and we have $WRAcc(d, l_2) = \frac{4}{6}(\frac{3}{4} - \frac{1}{2}) = \frac{1}{6}$. The upper part of the search space (most general subgroups) is given in Figure 4.2. The direct specializations of a subgroup are given, for each attribute, by adding a restriction: Either by shrinking the interval of values to the left (take the right next value in its domain) or to the right (take the left next value). In this way, the finite set of all intervals taking borders in the attributes domain will be explored (see [90]).*

Pattern set discovery consists in searching for a set of patterns $\mathcal{R} \subseteq \mathcal{S}$ of high quality on the quality measure φ and whose patterns are not redundant. As similar patterns generally have similar values on φ, we design the pattern set discovery problem as the identification of the local optima w.r.t. φ. As explained below, this has two main advantages: Redundant patterns of lower quality on φ are pruned and the extracted local optima are diverse and potentially interesting patterns.

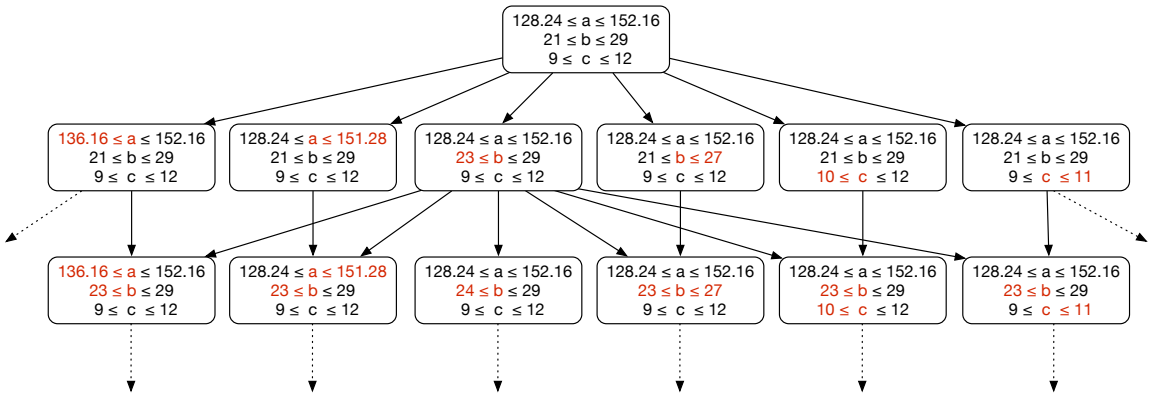


Figure 4.2: The upper part of the search space for Table 4.1.

Definition 29 (Local optimum as a non redundant pattern) *Let* $sim : \mathcal{S} \times \mathcal{S} \rightarrow [0,1]$ *be a similarity measure on* $\mathcal{S}$ *that, given a real value* $\Theta > 0$*, defines neighborhoods on* $\mathcal{R} \subseteq \mathcal{S} : N_{\mathcal{R}}(x) = \{s \in \mathcal{R} \mid sim(x,s) \geq \Theta\}$*.* $r^\star$ *is a local optimum of* $\mathcal{R}$ *on* φ *iff* $\forall r \in N_{\mathcal{R}}(r^\star)$, $\varphi(r^\star) \geq \varphi(r)$*. We denote by* $filter(\mathcal{R})$ *the set of local optima of* $\mathcal{R}$ *and by* $redundancy(\mathcal{R}) = 1 - \frac{|filter(\mathcal{R})|}{|\mathcal{R}|}$ *the measure of redundancy of* $\mathcal{R}$*.*

In this paper, the similarity measure on $\mathcal{S}$ will be the Jaccard measure defined by $sim(r,r') = \frac{ext(r) \cap ext(r')}{ext(r) \cup ext(r')}$.

We propose to evaluate the diversity of a pattern set $\mathcal{R} \subseteq \mathcal{S}$ by the sum of the quality of its patterns. Indeed, the objective is to obtain the largest set of high quality patterns:

Definition 30 (Pattern set diversity) *The diversity of a pattern set* $\mathcal{R}$ *is evaluated by:* $diversity(\mathcal{R}) = \sum_{r \in filter(\mathcal{R})} \varphi(r)$.

The function $filter()$ is generally defined in a greedy or heuristic way in the literature. [159] called it pattern set selection and we use their implementation in this article. First all extracted patterns are sorted according to the quality measure and the best one is kept. The next patterns in the order are discarded if they are too similar with the best pattern (a similarity function, here, a Jaccard between the pattern support is used). Once a non similar pattern is found, it is kept for the final result and the process is reiterated: Following patterns will be compared to it.

Problem 1 (Pattern set discovery) *Compute a set of patterns* $\mathcal{R}^* \subseteq \mathcal{S}$ *such that* $\forall r \in \mathcal{R}^*$, r *is a local optimum on* φ *and*

$$\mathcal{R}^* = argmax_{\mathcal{R} \subseteq \mathcal{S}} diversity(\mathcal{R}).$$

By construction, $\mathcal{R}^*$ maximizes diversity and it minimizes redundancy. Naturally, $\mathcal{R}^*$ is not unique. Existing approaches sometimes search for a pattern set of size k [110], with a minimum support threshold $minSupp$ [15].

4.3 Monte Carlo Tree Search

MCTS is a search method used in several domains to find an optimal decision (see the survey by [36]). It merges theoretical results from decision theory [142], game theory, Monte Carlo [2] and bandit-based methods [16]. MCTS is a powerful method because it enables the use of random simulations for characterizing a trade-off between the exploration of the search tree and the exploitation of an interesting solution, based on past observations. Considering a two-players game (e.g., Go): The goal of MCTS is to find the best action to play given a current game state. MCTS proceeds in several (limited) iterations

that build a partial game tree (called the search tree) depending on the results of previous iterations. The nodes represent game states. The root node is the current game state. The children of a node are the game states accessible from this node by playing an available action. The leaves are the terminal game states (game win/loss/tie). Each iteration, consisting of 4 steps (see Figure 4.3), leads to the generation of a new node in the search tree (depending on the exploration/exploitation trade-off due to the past iterations) followed by a simulation (sequence of actions up to a terminal node). Any node s in the search tree is provided with two values: The number $N(s)$ of times it has been visited, and a value $Q(s)$ that corresponds to the aggregation of rewards of all simulations walked through s so far (e.g., the proportion of wins obtained for all simulations walked through s). The aggregated reward of each node is updated through the iterations such that it becomes more and more accurate. Once the computation budget is reached, MCTS returns the best move that leads to the child of the root node with the best aggregated reward $Q(.)$.

In the following, we detail the 4 steps of a MCTS iteration applied to a game. Algorithm 5 gives the pseudo code of the most popular algorithm in the MCTS family, namely UCT (upper confidence bound for trees), as given in [95].

The Select policy. Starting from the root node, the SELECT method recursively selects an action (an edge) until the selected node is either a terminal game state or is not fully expanded (there remain children of this node that are not yet expanded in the search tree). The selection of a child of a node s is based on the exploration/exploitation trade-off. For that, upper confidence bounds (UCB) are used. They bound the regret of choosing a non-optimal child. The original UCBs used in MCTS are the UCB1 from [16] and the UCT from [95]:

$$UCT(s, s') = Q(s') + 2C_p\sqrt{\frac{2\ln N(s)}{N(s')}}$$

where s' is a child of a node s and $C_p > 0$ is a constant (generally, $C_p = \frac{1}{\sqrt{2}}$). This step selects the most urgent node to be expanded, called s_{sel} in the following, considering both the exploitation of interesting actions (given by the first term in UCT) and the exploration of lightly explored areas of the search space (given by the second term in UCT) based on the result of past iterations. The constant C_p can be adjusted to lower or increase the exploration weight in the exploration/exploitation trade-off . Note that when $C_p = \frac{1}{2}$, the UCT is called UCB1.

The Expand policy. A new child, denoted s_{exp}, of the selected node s_{sel} is added to the tree according to the available actions. The child s_{exp} is randomly picked among all available children of s_{sel} not yet expanded in the search tree.

The RollOut policy. From this expanded node s_{exp}, a simulation is played based on a specific policy. This simulation consists of exploring the search tree (playing a sequence of actions) from s_{exp} until a terminal state is reached. It returns the reward Δ of this terminal state: $\Delta = 1$ if the terminal state is a win, $\Delta = 0$ otherwise.

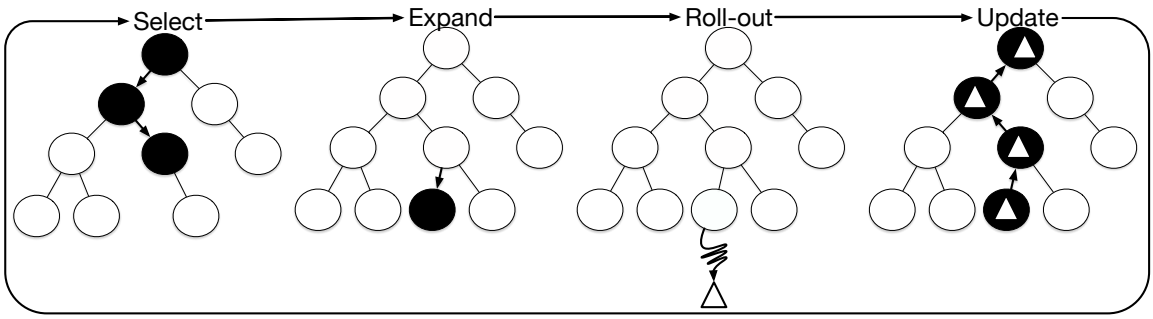


Figure 4.3: One MCTS iteration (taken from [36]).

Algorithm 5 UCT: The popular MCTS algorithm.

function MCTS(*budget*)
  create root node s_0 for current state
  while within computational budget *budget* **do**
    $s_{sel} \leftarrow$ SELECT(s_0)
    $s_{exp} \leftarrow$ EXPAND(s_{sel})
    $\Delta \leftarrow$ ROLLOUT(s_{exp})
    UPDATE(s_{exp}, Δ)
  end while
  return the action that reaches the child s of s_0 with the highest $Q(s)$
end function

function SELECT(s)
  while s is non-terminal **do**
    if s is not fully expanded **then** **return** s
    else $s \leftarrow$ BESTCHILD(s)
    end if
  end while
  return s
end function

function EXPAND(s_{sel})
  randomly choose s_{exp} from non expanded children of s_{sel}
  add new child s_{exp} to s_{sel}
  return s_{exp}
end function

function ROLLOUT(s)
  $\Delta \leftarrow 0$
  while s is non-terminal **do**
    choose randomly a child s' of s
    $s \leftarrow s'$
  end while
  return the reward of the terminal state s
end function

function UPDATE(s, Δ)
  while s is not null **do**
    $Q(s) \leftarrow \frac{N(s) \times Q(s) + \Delta}{N(s)+1}$
    $N(s) \leftarrow N(s) + 1$
    $s \leftarrow$ parent of s
  end while
end function

function BESTCHILD(s)
  return $\underset{s' \in \ children\ of\ s}{\arg\max} \ UCB(s, s')$
end function

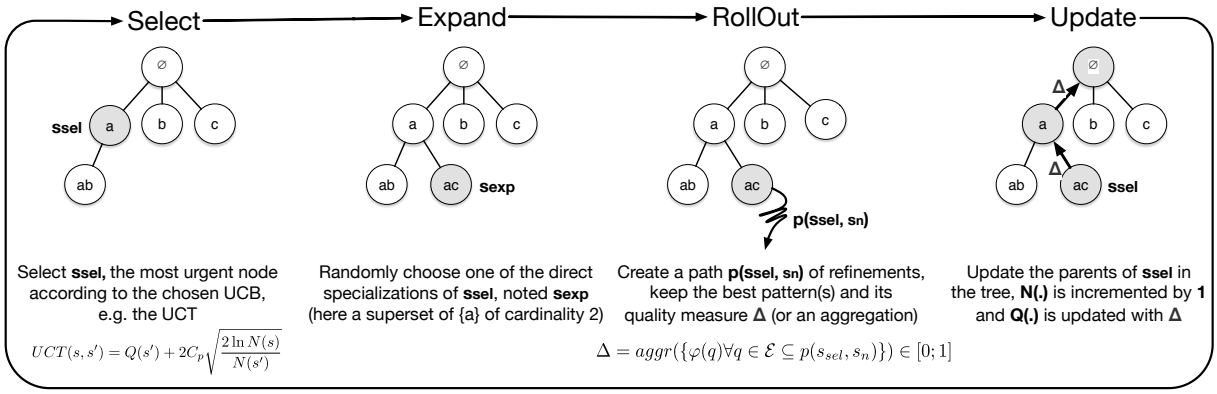


Figure 4.4: A simple instanciation of MCTS for pattern mining.

The Update policy. The reward Δ is back-propagated to the root, updating for each parent the number of visits $N(.)$ (incremented by 1) and the aggregation reward $Q(.)$ (the new proportion of wins).

Example 15 *Figure 4.3 depicts a MCTS iteration. Each node has no more than 2 children. In this scenario, the search tree is already expanded: We consider the* 9^{th} *iteration since* 8 *nodes of the tree have been already added. The first step consists in running the* Select *method starting from the root node. Based on a UCB, the* selection policy *chooses the left child of the root. As this node is fully expanded, the algorithm selects a new node among the children of this node: Its right child. This selected node* s_{sel} *is not fully expanded since its left hand side child is not in the search tree yet. From this not fully expanded node* s_{sel}*, the* Expand *method adds the left hand side child* s_{exp} *of the selected node* s_{sel} *to expand the search tree. From this added node* s_{exp}*, a simulation is rolled out until reaching a terminal state. The reward* Δ *of the terminal node is back-propagated with* Update*.*

4.4 Pattern Set Discovery with MCTS

Designing a MCTS approach for a pattern mining problem is different than for a combinatorial game: The goal is not to decide, at each turn, what is the best action to play, but to explore the search space: This can be considered as a *single-turn single-player game*. Most importantly, MCTS offers a natural way to explore the search space of patterns with the benefit of the exploitation/exploration trade-off to improve diversity while limiting redundancy. For example, an exhaustive search will maximize diversity, but it will return a very large and redundant collection (if it runs). In contrast, a beam search can extract a limited number of patterns but it will certainly lack diversity (we give empirical evidences in our experimental results).

Before going into the formalization, let us illustrate how MCTS is applied to the pattern set discovery problem with Figure 4.4. We consider here itemset patterns for the sake of simplicity, that is, subgroups whose descriptions are sets of items. We present an iteration of a MCTS for a transaction database with items $\mathcal{I} = \{a, b, c\}$. The pattern search space is given by the lattice $\mathcal{S} = (2^{\mathcal{I}}, \subseteq)$. The MCTS tree is built in a top-down fashion on this theoretical search space: The initial pattern, or root of the tree, is the empty set $\emptyset$. Assume that pattern $s_{sel} = \{a\}$ has been chosen by the *select* policy. During the *expand*, one of its direct specializations in $\{\{a, b\}, \{a, c\}\}$ is randomly chosen and added to the tree,e.g., $s_{exp} = \{a, c\}$. During the roll out, a simulation is run from this node: it generates a chain of specializations of s_{exp} called a path $p(s_{sel}, s_n)$ (a chain is a set of comparable patterns w.r.t. $\subseteq$, or $\preceq$ in the general case). The quality measure φ is computed for each pattern of the path, and an aggregated value (max, mean, etc.) is returned and called Δ. Finally, all parents of s_{exp} are updated: Their visit count $N(.)$ is incremented by one, while their quality estimation $Q(.)$ is recomputed with Δ (back propagation). The new values of $N(.)$ and $Q(.)$ will directly impact the selection of the next iteration when computing the chosen UCB,

Select
Choose one of the following UCB: **UCB1** or **UCB1-Tuned** or **SP-MCTS** or **UCT**
Expand
direct-expand: Randomly choose the next direct expansion **gen-expand**: Randomly choose the next direct expansion until it changes the extent **label-expand**: Randomly choose the next direct expansion until it changes the true positives **Activate LO**: Generate each pattern only once (lectic enumeration) **Activate PU**: Patterns with the same support/true positive set point to the same node
RollOut
naive-roll-out: Generate a random path of direct specializations of random length. **direct-freq-roll-out**: Generate a random path of frequent direct specializations. **large-freq-roll-out**: Generate a random paths of undirect specializations (random jumps).
Memory
no-memory: No pattern found during the simulation is kept for the final result. **top-k-memory**: Top-k patterns of a simulation are considered in memory. **all-memory**: All patterns generated during the simulation are kept.
Update
max-update: Only the maximum φ found in a simulation is back propagated **mean-update**: The average of all φ is back-propagated **top-k-mean-update**: The average of the best k φ is back-propagated

Table 4.2: The different policies

and thus the desired exploration/exploitation trade off. When the budget is exceeded (or if the tree is fully expanded), all patterns are filtered with a chosen pattern set selection strategy ($filter(.)$).

The expected shape of the MCTS tree after a high number of iterations is illustrated in Figure 4.1d. It suggests a high diversity of the final pattern set if given enough budget (i.e., enough iterations). However, how to properly define each policy (select, expand, roll out and update), is not obvious. Table 4.2 sums up the different policies that we use or develop specifically for a pattern mining problem.

4.4.1 The Select method

The SELECT method has to select the most promising node s_{sel} in terms of the exploration vs. exploitation trade-off. For that, the well-known bounds like UCT or UCB1 can be used. However, more sophisticated bounds have been designed for single player games. The single-player MCTS (SP-MCTS), introduced by [144], adds a third term to the UCB to take into account the variance σ^2 of the rewards obtained by the child so far. SP-MCTS of a child s' of a node s is:

$$SP\text{-}MCTS(s, s') = Q(s') + C\sqrt{\frac{2\ln N(s)}{N(s')}} + \sqrt{\sigma^2(s') + \frac{D}{N(s')}}$$

where the constant C is used to weight the exploration term (it is fixed to 0.5 in its original definition) and the term $\frac{D}{N(s')}$ inflates the standard deviation for infrequently visited children (D is also a constant). In this way, the reward of a node rarely visited is considered as less certain: It is still required to explore it to get a more precise estimate of its variance. If the variance is still high, it means that the subspace from this node is not homogeneous w.r.t. the quality measure and further exploration is needed.

Also, [16] designed *UCB1-Tuned* to reduce the impact of the exploration term of the original UCB1 by weighting it with either an approximation of the variance of the rewards obtained so far or the factor

1/4. UCB1-Tuned of a child s' of s is:

$$UCB1\text{-}Tuned(s, s') = Q(s') + \sqrt{\frac{\ln N(s)}{N(s')} \min(\frac{1}{4}, \sigma^2(s') + \sqrt{\frac{2 \ln N(s)}{N(s')}})}$$

The only requirement the pattern quality measure φ must satisfy is, in case of UCT only, to take values in $[0, 1]$: φ can be normalized in this case.

4.4.2 The Expand method

The EXPAND step consists in adding a pattern specialization as a new node in the search tree. In the following, we present different refinement operators, and how to avoid duplicate nodes in the search tree.

4.4.2.1 The refinement operators

A simple way to expand the selected node s_{sel} is to choose uniformly an available attribute w.r.t. s_{sel}, that is to specialize s_{sel} into s_{exp} such that $s_{exp} \prec s_{sel}$: s_{exp} is a refinement of s_{sel}. It follows that $ext(s_{exp}) \subseteq ext(s_{sel})$, and obviously $supp(s_{exp}) \leq supp(s_{sel})$, known as the monotonocity property of the support.

Definition 31 (Refinement operator) *A refinement operator is a function $ref : \mathcal{S} \to 2^{\mathcal{S}}$ that derives from a pattern s a set of more specific patterns $ref(s)$ such that:*

(i) $\forall s' \in ref(s), s' \prec s$

(ii) $\forall s'_i, s'_j \in ref(s), i \neq j, s'_i \not\preceq s'_j, s'_j \not\preceq s'_i$

In other words, a refinement operator gives to any pattern s a set of its specializations, that are pairwise incomparable (an anti-chain). The *refine* operation can be implemented in various ways given the kind of patterns we are dealing with. Most importantly, it can return all the direct specializations only to ensure that the exploration will, if given enough budget, explore the whole search space of patterns. Furthermore, it is unnecessary to generate infrequent patterns.

Definition 32 (Direct-refinement operator) *A direct refinement operator is a refinement operator $directRef : \mathcal{S} \to 2^{\mathcal{S}}$ that derives from a pattern s the set of direct more specific patterns s' such that:*

(i) $\forall s' \in directRef(s)$, $s' \prec s$

(ii) $\nexists s'' \in \mathcal{S}$ *s.t.* $s' \prec s'' \prec s$

(iii) For any $s' \in directRef(s)$, s' *is frequent, that is* $supp(s') \geq minSupp$

The notion of direct refinement is well known in pattern mining. For instance, the only way to refine a nominal (resp. Boolean) attribute is to assign it a value of its domain (resp. the *true* value). Refining an itemset consists in adding a item, while refining a numerical attribute can be done in two ways: Applying the minimal left change (resp. right change), that is, increasing the lower bound of the interval to the next higher value in its domain (resp. decreasing the upper bound to the next lower) as explained by [90]. We still use the term *restriction* to denote the operations that create a direct refinement of pattern.

Definition 33 (The *direct-expand* strategy) *We define the direct-expand strategy as follows: From the selected node s_{sel}, we randomly pick a – not yet expanded – node s_{exp} from $directRef(s_{sel})$ and add it in the search tree.*

As most quality measures φ used in SD and EMM are solely based on the extent of the patterns, considering only one pattern among all those having the same extent is enough. However, with the *direct-refinement operator*, a large number of tree nodes may have the same extent as their parent. This redundancy may bias the exploration and more iterations will be required. For that, we propose to use the notion of closed patterns and their generators.

Definition 34 (Closed descriptions and their generators) *The equivalence class of a pattern s is given by $[s] = \{s' \in \mathcal{S} \mid ext(s) = ext(s')\}$. Each equivalence class has a unique smallest element w.r.t. $\prec$ that is called the closed pattern: s is said to be closed iff $\nexists s'$ such that $s' \prec s$ and $ext(s) = ext(s')$. The non-closed patterns are called generators.*

Definition 35 (Generator-refinement operator) *A generator refinement operator is a refinement operator $genRef : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ that derives from a pattern s the set of more specific patterns s' such that, $\forall s' \in genRef(s)$:*

(i) $s' \notin [s]$ (different support)

(ii) $\nexists s'' \in \mathcal{S} \backslash genRef(s)$ s.t. $s'' \notin [s]$, $s'' \notin [s']$, $s' \prec s'' \prec s$ (direct next equivalence class)

(iii) s' is frequent, that is $supp(s') \geq minSupp$ (frequent)

Definition 36 (The *gen-expand* strategy) *To avoid the exploration of patterns with the same extent in a branch of the tree, we define the min-gen-expand strategy as follows: From the selected node s_{sel}, we randomly pick a – not yet expanded – refined pattern from $genRef(s_{sel})$, called s_{exp}, and add it to the search tree.*

Finally, when facing a SD problem whose aim is to characterize a label $l \in \mathcal{C}$ we can adapt the previous refinement operator based on generators on the extents of both the subgroup and the label. As many other measures, the WRAcc seeks to optimize the (weighted relative) precision or accuracy of the subgroup. The accuracy is the ratio of true positives in the extent. We propose thus, for this kind of measures only, the ***label-expand*** strategy: Basically, the pattern is refined until the set of true positives in the extent changes. This minor improvement performs very well in practice as pointed our in our experimental evaluation.

4.4.2.2 Avoiding duplicates in the search tree

We defined several refinement operators to avoid the redundancy within a branch of the tree, i.e., do not expand s_{sel} with a pattern whose extent is the same because the quality measure φ will be equal. However, another redundancy issue remains at the tree scale. Indeed, since the pattern search space is a lattice, a pattern can be generated in nodes from different branches of the Monte Carlo tree, that is, with different sequences of refinements, or simply permutations of refinements. As such, it will happen that a part of the search space is sampled several times in different branches of the tree. However, the visit count $N(s)$ of a node s will not count visits of other nodes that denote exactly the same pattern: The UCB is clearly biased. To tackle this aspect, we implement two methods: (i) Using a lectic order or (ii) detecting and unifying the duplicates within the tree. These two solutions can be used for any refinement operator. Note that enabling both these solutions at the same tame is useless since each of them ensures to avoid duplicates within the tree.

Avoiding duplicates in the tree using a lectic order (LO).

Pattern enumeration without duplicates is at the core of constraint-based pattern-mining [[33]]. Avoiding to generate patterns with the same extent is usually based on a total order on the set of attribute restrictions. This poset is written by $(R, \preccurlyeq)$.

Example 16 *For instance, considering itemset patterns, $R = \mathcal{I}$ and a lectic order, usually the lexicographic order, is chosen on $\mathcal{I}$: $a \lessdot b \lessdot c \lessdot d$ for $I = \{a, b, c, d\}$ and $bc \lessdot ad$. Consider that a node s has been generated with a restriction r_i: we can expand the node only with restrictions r_j such that $r_i \lessdot r_j$. This total order also holds for numerical attributes by considering the minimal changes (see the work of [90] for further details).*

We can use this technique to enumerate the lattice with a depth-first search (DFS), which ensures that each element of the search space is visited exactly once. An example is given in Figure 4.5. However, it induces a strong bias: An MCTS algorithm would sample this tree instead of sampling the pattern search space. In other words, a small restriction w.r.t. $\lessdot$ has much less chances to be picked than a largest one. Going back to the example in Figure 4.5 (middle), the item a can be drawn only once through a complete DFS; b twice; while c four times (in bold). It follows that patterns on the left hand side of the tree have less chances to be generated, e.g., $prob(\{a, b\}) = 1/6$ while $prob(\{b, c\}) = 1/3$. These two itemsets should however have the same chance to be picked as they have the same size. This disequilibrium can be corrected by weighting the visit counts in the UCT with the normalized exploration rate (see Figure 4.5 (right)).

Definition 37 (Normalized exploration rate) *Let $\mathcal{S}$ be the set of all possible patterns. The normalized exploration rate of a pattern s is,*

$$\rho_{norm}(s) = \frac{V_{total}(s)}{V_{lectic}(s)} = \frac{|\{s' | s' \preceq s, \forall s' \in \mathcal{S}\}|}{|\{s' | (s \lessdot s' \wedge s' \prec s) \vee s = s', \forall s' \in \mathcal{S}\}|}$$

Given this normalized exploration rate, we can adapt the UCBs when enabling the lectic order. For example, we can define the *DFS-UCT* of a child s' of a pattern s derived from the *UCT* as follows:

$$DFS\text{-}UCT(s, s') = Q(s') + 2C_p \sqrt{\frac{2 \ln \left(N(s) \cdot \rho_{norm}(s)\right)}{N(s') \cdot \rho_{norm}(s')}}$$

Proposition 11 (Normalized exploration rate for itemsets) *For itemsets, let s_i be the child of s obtained by playing action r_i and i is the rank of r_i in $(R, \lessdot)$: $\rho_{norm}(s_i) = \frac{2^{(|\mathcal{I}|-|s_i|)}}{2^{(|\mathcal{I}|-i-1)}}$.*

Proof 7 *Let $V_{lectic}(s_i)$ be the size of the search space sampled under s_i using a lectic enumeration, and $V_{total}(s_i)$ be the size of the search space without using a lectic enumeration. Noting $V_{total}(s_i) = 2^{(|\mathcal{I}|-|s_i|)}$ and $V_{lectic}(s_i) = 2^{(|\mathcal{I}|-i-1)}$ for itemsets, we have $\rho_{norm}(s_i) = \frac{V_{total}(s_i)}{V_{lectic}(s_i)} = \frac{2^{(|\mathcal{I}|-|s_i|)}}{2^{(|\mathcal{I}|-i-1)}}$.*

Proposition 12 (Normalized exploration rate for a numerical attribute) *For a single numerical attribute a, $\rho_{norm}(.)$ is defined as follows :*

- *Let $s' = \langle \alpha_i \leq a \leq \alpha_j \rangle$ obtained after a left change: $\rho_{norm}(s') = 1$.*
- *Let $s' = \langle \alpha_i \leq a \leq \alpha_j \rangle$ obtained after a right change. Let n be the number of values from $Dom(a)$ in $[\alpha_i, \alpha_j]$: $\rho_{norm}(s') = \frac{n+1}{2}$.*

Proof 8 *As explained in the proof of (Proposition 11), $\rho_{norm}(s) = \frac{V_{total}(s)}{V_{lectic}(s)}$. For a numerical attribute, $V_{total}(s) = n(n+1)/2$, i.e. the number of all sub intervals. If s was obtained after a left change, $V_{lectic}(s) = n(n+1)/2$ as both left and right changes can be applied. If s was obtained after a right change, $V_{lectic}(s) = n$, as only n right changes can be applied. It follows that $\rho_{norm}(s) = \frac{n(n+1)/2}{n(n+1)/2} = 1$ if s was obtained from a left change and $\rho_{norm}(s) = \frac{n(n+1)/2}{n} = \frac{n+1}{2}$ otherwise.*

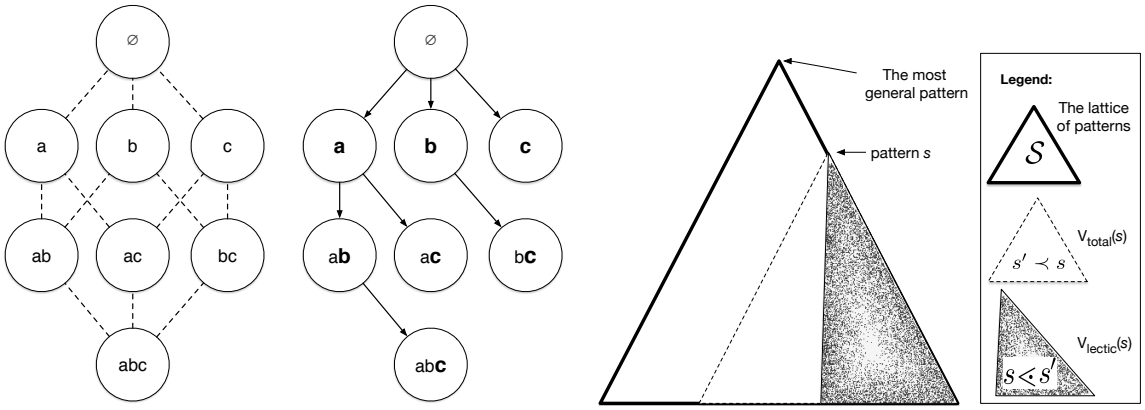


Figure 4.5: Search space as a lattice (left), DFS of the search space (middle), and the principles of the normalized exploration rate.

Avoiding duplicates in the tree using permutation unification (PU).

The permutation unification is a solution that enables to keep a unique node for all duplicates of a pattern that can be expanded within several branches of the tree. This is inspired from *Permutation AMAF* of [78], a method used in traditional MCTS algorithms to update all the nodes that can be concerned by a play-out. A unified node no longer has a single parent but a list of all duplicates' parent. This list will be used when back-propagating a reward.

This is detailed in Algorithm 6. Consider that the node s_{exp} has been chosen as an expansion of the selected node s_{sel}. The tree generated so far is explored for finding s_{exp} elsewhere in the tree: If s_{exp} is not found, we proceed as usual; otherwise s_{exp} becomes a pointer to the duplicate node in the tree. In our MCTS implementation, we will simply use a hash map to store each pattern and the node in which is has been firstly encountered.

Algorithm 6 The permutation unification principle.

```
1: H ← new Hashmap()
2: function EXPAND(s_sel)
3:     randomly choose s_exp from non expanded children of s_sel
4:     if (node ← H.get(s_exp)) ≠ null then
5:         node.parents.add(s_sel)
6:         s_exp ← node
7:     else
8:         s_exp.parents ← new List()
9:         s_exp.parents.add(s_sel)
10:        H.put(s_exp, s_exp)                ▷ A pointer on the unique occurrence of s_exp
11: end if
12: add new child s_exp to s_sel in the tree  ▷ Expand s_sel with s_exp
13: return s_exp
14: end function
```

4.4.3 The RollOut method

From the expanded node s_{exp} a simulation is run (RollOut). With standard MCTS, a simulation is a random sequence of actions that leads to a terminal node: A game state from which a reward can be computed (win/loss). In our settings, it is not only the leaves that can be evaluated, but any pattern s encountered during the simulation. Thus, we propose to define the notion of path (the simulation)

and reward computation (which nodes are evaluated and how these different rewards are aggregated) separately.

Definition 38 (Path policy) *Let s_1 the node from which a simulation has to be run (i.e., $s_1 = s_{exp}$). Let $n \geq 1 \in \mathbb{N}$, we define a path $p(s_1, s_n) = \{s_1, \ldots, s_n\}$ as a chain in the lattice $(\mathcal{S}, \prec)$, i.e., an ordered list of patterns starting from s_1 and ending with s_n such that $\forall i \in \{1, \ldots, n-1\}, s_{i+1}$ is a (not necessarily direct) refined pattern of s_i.*

- *naive-roll-out: a path of direct refinements is randomly created with length $pathLength \in \mathbb{N}^+$ a user-defined parameter.*
- *direct-freq-roll-out: The path is extended with a randomly chosen restriction until it meets an infrequent pattern s_{n+1} using the direct refinement operator. Pattern s_n is a leaf of the tree in our settings.*
- *large-freq-roll-out overrides the direct-freq-roll-out policy by using specializations that are not necessarily direct. Several actions are added instead of one to create a new element of the path. The number of added actions is randomly picked in $(1, ..., jumpLength)$ where $jumpLength$ is given by the user ($jumpLength = 1$ gives the previous policy). This techniques allows to visit deep parts of the search space with shorter paths.*

Definition 39 (Reward aggregation policy) *Let s_1 be the node from which a simulation has been run and $p(s_1, s_n)$ the associated random path. Let $\mathcal{E} \subseteq p(s_1, s_n)$ be the subset of nodes to be evaluated. The aggregated reward of the simulation is given by: $\Delta = aggr(\{\varphi(s) \forall s \in \mathcal{E}\}) \in [0; 1]$ where aggr is an aggregation function. We define several reward aggregation policies:*

- *terminal-reward: $\mathcal{E} = \{s_n\}$ and aggr is the identity function.*
- *random-reward: $\mathcal{E} = \{s_i\}$ with a random $1 \leq i \leq n$ and aggr is the identity function.*
- *max-reward: $\mathcal{E} = p(s_1, s_n)$ and aggr is the $max(.)$ function*
- *mean-reward: $\mathcal{E} = p(s_1, s_n)$ and aggr is the $mean(.)$ function.*
- *top-k-mean-reward: $\mathcal{E} = top\text{-}k(p(s_1, s_n))$, aggr is the $mean(.)$ function and top-k(.) returns the k elements with the highest φ.*

A basic MCTS forgets any state encountered during a simulation. This is not optimal for single player games as relate [26]: A pattern with a high φ should not be forgotten as we might not expand the tree enough to reach it. We propose to consider several memory strategies.

Definition 40 (Roll-out memory policy) *A roll-out memory policy specifies which of the nodes of the path $p = (s_1, s_n)$ shall be kept in an auxiliary data structure M.*

- *no-memory: Any pattern in $\mathcal{E}$ is forgotten.*
- *all-memory: All evaluated patterns in $\mathcal{E}$ are kept.*
- *top-k-memory: A list M stores the best k patterns in $\mathcal{E}$ w.r.t. $\varphi(.)$.*

This structure M will be used to produce the final pattern set.

4.4.4 The Update method

The backpropagation method updates the tree according to a simulation. Let s_{sel} be the selected node and s_{exp} its expansion from which the simulation is run: This step aims at updating the estimation $Q(.)$ and the number of visits $N(.)$ of each parent of s_{exp} recursively. Note that s_{exp} may have several parents when we enable permutation unification (PU). The number of visits is always incremented by one. We consider three ways of updating $Q(.)$:

- *mean-update*: $Q(.)$ is the average of the rewards Δ back-propagated through the node so far (basic MCTS).
- *max-update*: $Q(.)$ is the maximum reward Δ back-propagated through the node so far. This strategy enables to identify a local optimum within a part of the search space that contains mostly of uninteresting patterns. Thus, it gives more chance for this area to be exploited in the next iterations.
- *top-k-mean-update*: $Q(.)$ average of the k best rewards Δ back-propagated through the node so far. It gives a stronger impact for the parts of the search space containing several local optima.

mean-update is a standard in MCTS techniques. We introduce the *max-update* and *top-k-mean-update* policies as it may often happen that high-quality patterns are rare and scattered in the search space. The mean value of rewards from simulations would converge towards 0 (there are too many low quality subgroups), whereas the maximum value (and top-k average) of rewards enables to identify the promising parts of the search space.

4.4.5 Search end and result output

There are two ways a MCTS ends: Either the computational budget is reached (number of iterations) or the tree is fully expanded (an exhaustive search has been possible, basically when the size of the search space is smaller than the number of iterations). Indeed, the number of tree nodes equals the number of iterations that have been performed. It remains now to explore this tree and the data structure M built by the memory policy to output the list of diverse and non-redundant patterns.

Let $\mathcal{P} = T \cup M$ be a pool of patterns, where T is the set of patterns stored in the nodes of the tree. The set $\mathcal{P}$ is totally sorted w.r.t. φ in a list Λ. Thus, we have to pick the k-best diverse and non-redundant subgroups within this large pool of nodes Λ to return the result set of subgroups $\mathcal{R} \subseteq \mathcal{P}$. For that, we choose to implement $filter(.)$ in a greedy manner as done by[159, 31]. $\mathcal{R} = filter(\mathcal{P})$ as follows: A post-processing that filters out redundant subgroups from the diverse pool of patterns Λ based on the similarity measure sim and the maximum similarity threshold Θ. Recursively, we poll (and remove) the best subgroup s^* from Λ, and we add s^* to $\mathcal{R}$ if it is not redundant with any subgroup in $\mathcal{R}$. It can be shown easily that $redundancy(\mathcal{R}) = 0$.

Applying $filter(.)$ at the end of the search requires however that the pool of patterns $\mathcal{P}$ has a reasonable cardinality which may be problematic with MCTS in term of memory. The allowed budget always enables such post-processing in our experiments (up to one million iterations).

4.5 Conclusion

Our MCTS implementation for pattern mining, called MCTS4DM is publicly available[5]. We end this chapter by a summary of a deep empirical study reported [30]

First, we experimented with the several strategies we defined for our algorithm MCTS4DM. Our recommendations are the following:

[5] `https://github.com/guillaume-bosc/MCTS4DM`

- SELECT: Concerning the choice of the upper confidence bound, it seems more suitable to use the *SP-MCTS* for SD problems, although it has a limited impact. Activating LO leads to worse results, but with PU we are able to get more interesting patterns. This is a quite interesting fact as LO is a widely used technique in pattern mining (enumerate each pattern only once with a lectic order).
- EXPAND: We advise to use the *label-gen* strategy that enables to reach more quickly the best patterns, but it can require more computational time.
- ROLLOUT: For nominal attributes, the *direct-freq-roll-out* is an efficient strategy. However, when facing numerical attributes, we recommend to employ the *large-freq-roll-out* since it may require a lot of time to reach the maximal frequent patterns.
- MEMORY: Using a memory strategy is essential since it enables to store the patterns encountered during the ROLLOUT step. The *top-1-memory* is enough to avoid to miss interesting patterns that are located deeper in the search space.
- UPDATE: When there are potentially many local optima in the search space, we recommend to set the *mean-update* strategy for the UPDATE step. Indeed it enables to exploit the areas that are deemed to be interesting in average. However, when there are few local optima among lots of uninteresting patterns, using *mean-update* is not optimal since the mean of the rewards would converge to 0. In place, the *max-update* should be used to ensure that an area containing a local optima is well identified.

Our second batch of experiments compared MCTS4DM with the main existing approaches for SD. For that, we experimented with one of the most efficient exhaustive search in SD, namely SD-MAP*, a beam search, the recent evolutionary algorithm SSDP and a sampling method implemented in the algorithm MISERE. The results suggest that MCTS4DM leads, in general, to a more diverse result set when an exhaustive search is not tractable. The greedy property of the beam search leads to a low diversity in the result set, and the lack of memory in sampling methods avoid to exploit interesting patterns to find the local optima (a pattern may be drawn several times). There is no guarantee that evolutionary algorithms and sampling approaches converge to the optimal pattern set even with an infinite computational budget.

MCTS comes with several advantages but has some limits:

+ It produces a good pattern set anytime and it converges to an exhaustive search if given enough time and memory (a *best-first search*).
+ It is agnostic of the pattern language and the quality measures: It handles numerical patterns without discretization in a pre-processing step and it still provides a high diversity using several quality measures.
+ MCTS4DM is aheuristic: No hypotheses are required to run the algorithm whereas with some sampling methods, a probability distribution (based on the quality measure and the pattern space) has to be given as a parameter.
- MCTS4DM may require a lot of memory. This memory usage becomes more and more important with the increase of the number of iterations.
- Despite the use of UCB, it is now well known that MCTS algorithms explore too much the search space. As MCTS basically requires to expand all the children of a node before exploiting one of them, this problem is even stronger when dealing with very high branching factor (number of direct specializations of a pattern). This problem has been in part tackled by the progressive widening approach that enables to exploit a child of a node before all of the other children of the node have been expanded [67, 36].

Heuristic search of supervised patterns becomes mandatory with large datasets. However, classical heuristics lead to a weak diversity in pattern sets: Only few local optima are found. We advocate for

the use of MCTS for pattern mining: An exploration strategy leading to *"any-time"* pattern mining that can be adapted with different measures and policies. The experiments show that MCTS provides a much better diversity in the result set than existing heuristic approaches. For instance, interesting subgroups are found by means of a reasonable amount of iterations and the quality of the result iteratively improves.

MCTS is a powerful exploration strategy that can be applied to several, if not all, pattern mining problems that need to optimize a quality measure given a subset of objects. For example, in a previous chapter, we have already tuned MCTS4DM for mining convex polygon patterns in numerical data [22].

In general, the main difficulties are to be able to deal with large branching factors, and jointly deal with several quality measures. This opens new research perspectives for mining more complex patterns such as sequences and graphs.

We obtain a first result with sequential pattern mining. In that case, the pattern search space is such that we cannot even afford to sample it directly. Instead, we chose to sample the space of object sets: the method becomes (almost) language agnostic and support-based pattern quality measures are easier to compute. A preliminary result can be found in [115] where a (very) simple bandit technique is used to sample sequential subgroups with promising results.

Chapter 5

Anytime Subgroup Discovery in Numerical Domains with Guarantees

One of the main motivation of investigating MCTS for pattern discovery (presented in the previous chapter) was actually the difficulty, but attractiveness of considering the set of all interval patterns. It is attractive, as it guarantees us to find the best patterns. It is difficult because most of the existing approaches are based either on a greedy selection of the intervals, either on an exhaustive search that is not applicable on large dataset. MCTS offers us the guarantee to be exhaustive if enough budget is given, but one cannot expect a budget that is sufficient enough and has to interrupt the search at some point. At interruption, we do have no idea on how far we are from the optimal solution. Moreover, the interval pattern generation (shrinking direct left and right values) makes that we need to sample very deeply the search space so that interesting intervals can be found ("not degenerated, but not too large"). It results that the search tree can be extremely large at some point. We thus concentrated our effort (i) a better exploration strategy for numerical patterns, (ii) the search for guarantees that tell us, e.g., "how far are we from the end of the (exhaustive) search".

In this chapter, we present a first step. We design an algorithm for mining numerical data with three key properties w.r.t. the state of the art: (i) It yields progressively interval patterns whose quality improves over time; (ii) It can be interrupted anytime and always gives a guarantee bounding the *error* on the top pattern quality and (iii) It always bounds a distance to the exhaustive exploration. After reporting experimentations showing the effectiveness of our method, we discuss its generalization to other kinds of patterns.

5.1 Introduction

We address the problem of discovering patterns that accurately discriminate one class label from the others in a numerical dataset. Subgroup discovery (SD) [167] is a well established pattern mining framework which strives to find out data regions uncovering such interesting patterns. When it comes to numerical attributes, a pattern is generally a conjunction of restrictions over the attributes, e.g., pattern $50 \leq age < 70 \wedge smoke\_per\_day \geq 3$ fosters lung cancer incidence. To look for such patterns (namely interval patterns), various approaches are usually implemented. Common techniques perform *a discretization* transforming the numerical attributes to categorical ones in a pre-processing phase before using the wide spectrum of existing mining techniques [15, 159, 110, 28]. This leads, however, to a loss of information even if an exhaustive enumeration is performed on the transformed data [15]. Other approaches explore the whole search space of all restrictions either exhaustively [90, 73, 37] or heuristically [113, 30]. While an exhaustive enumeration is generally unfeasible in large data, the various state-of-the-art algorithms that heuristically explore the search space provide no provable guarantee on how they *approximate* the top quality patterns and on how far they are from an exhaustive search. Recent techniques set up a third and

elegant paradigm, that is direct sampling approaches [28, 29, 70]. Algorithms falling under this category are non-enumerative methods which directly sample solutions from the pattern space. They simulate a distribution which rewards high quality patterns with respect to some interestingness measure. While [28, 29] propose a direct two-step sampling procedure dedicated for categorical/boolean datasets, authors in [70] devise an interesting framework which add a third step to handle the specificity of numerical data. The proposed algorithm addresses the discovery of dense neighborhood patterns by defining a new density metric. Nevertheless, it does not consider the discovery of discriminant numerical patterns in labeled numerical datasets. Direct sampling approaches abandon the completeness property and generate only approximate results. In contrast, anytime pattern mining algorithms [30, 80] are enumerative methods which exhibits the anytime feature [174], a solution is always available whose quality improves gradually over time and which converges to an exhaustive search if given enough time, hence ensuring completeness. However, to the best of our knowledge, no existing anytime algorithm in SD framework, makes it possible to ensure guarantees on the patterns discriminative power and the remaining distance to an exhaustive search while taking into account the nature of numerical data.

To achieve this goal, we propose a novel anytime algorithm, `RefineAndMine`, tailored for discriminant interval patterns discovery in numerical data. It starts by mining interval patterns in a coarse discretization, followed by successive refinements yielding increasingly finer discretizations highlighting potentially new interesting patterns. Eventually, it performs an exhaustive search, if given enough time. Additionally, our method gives two provable guarantees at each refinement. The first evaluates how close is the best found pattern so far to the optimal one in the whole search space. The second measures how already found patterns are diverse and cover well all the interesting regions in the dataset.

The outline is as follows. We recall in Sec. 5.2 basic definitions. Next, we define formally the problem in Sec. 5.3. Subsequently We introduce in Sec. 5.4 our mining algorithm before formulating the guarantees it provides in Sec. 5.5. An empirical evaluation can be found in the original article. We discuss its potential improvements in Sec. 5.6. Additional materials are available in our companion page[6]. For more details, please refer to the supplementary material[7].

5.2 Preliminaries

Input. A *labeled numerical dataset* $(\mathcal{G}, \mathcal{M})$ is given by a finite set (of objects) $\mathcal{G}$ partitioned into two subsets $\mathcal{G}^+$ and $\mathcal{G}^-$ enclosing respectively positive (target) and negative instances; and a sequence of numerical attributes $\mathcal{M} = (m_i)_{1 \leq i \leq p}$ of size $p = |\mathcal{M}|$. Each *attribute* m_i is an application $m_i : \mathcal{G} \to \mathbb{R}$ that associates to each object $g \in \mathcal{G}$ a value $m_i(g) \in \mathbb{R}$. We can also see $\mathcal{M}$ as a mapping $\mathcal{M} : \mathcal{G} \to \mathbb{R}^p, g \mapsto (m_i(g))_{1 \leq i \leq p}$. We denote $m_i[\mathcal{G}] = \{m_i(g) \mid g \in \mathcal{G}\}$ (More generally, for a function $f : E \to F$ and a subset $A \subseteq E$, $f[A] = \{f(e) \mid e \in A\}$). Fig. 5.1 (left table) presents a 2-dimensional labeled numerical dataset and its representation in the Cartesian plane (filled dots represent positive instances).

Interval patterns and their extents. When dealing with numerical domains in SD, we generally consider for intelligibility *interval patterns* [90]. An *Interval pattern* is a conjunction of restrictions over the numerical attributes; i.e. a set of conditions *attribute* $\gtrless v$ with $\gtrless \in \{=, \leq, <, \geq, >\}$. Geometrically, interval patterns are *axis-parallel hyper-rectangles*. Fig. 5.1 (center-left) depicts pattern (non-hatched rectangle) $c_2 = (1 \leq m_1 \leq 4) \wedge (0 \leq m_2 \leq 3) \triangleq [1, 4] \times [0, 3]$.

Interval patterns are naturally partially ordered thanks to "hyper-rectangle inclusion". We denote the *infinite partially ordered set* (*poset*) of all interval patterns by $(\mathcal{D}, \sqsubseteq)$ where $\sqsubseteq$ (same order used in [90]) denotes the dual order $\supseteq$ of hyper-rectangle inclusion. That is pattern $d_1 \sqsubseteq d_2$ *iff* d_1 encloses d_2 ($d_1 \supseteq d_2$). It is worth mentioning that $(\mathcal{D}, \sqsubseteq)$ forms a *complete lattice* [141]. For a subset $S \subseteq \mathcal{D}$, the join $\bigsqcup S$ (i.e. smallest upper bound) is given by the rectangle intersection. Dually, the meet $\bigsqcap S$ (i.e the largest lower bound) is given by the smallest hyper-rectangle enclosing all patterns in S. Note that the top (resp. bottom) pattern in $(\mathcal{D}, \sqsubseteq)$ is given by $\top = \emptyset$ (resp. $\bot = \mathbb{R}^p$). Fig. 5.1 (right) depicts two

[6] `https://github.com/Adnene93/RefineAndMine`
[7] `https://goo.gl/NWtXfp`

patterns (hatched) $e_1 = [1,5]\times(1,4]$ and $e_2 = [0,4)\times[2,6]$, their meet (non hatched) $e_1 \sqcap e_2 = [0,5]\times(1,6]$ and their join (black) $e_1 \sqcup e_2 = [1,4)\times[2,4]$.

A pattern $d \in \mathcal{D}$ is said to cover an object $g \in \mathcal{G}$ iff $\mathcal{M}(g) \in d$. To use the same order $\sqsubseteq$ to define such a relationship, we associate to each $g \in \mathcal{G}$ its corresponding pattern $\delta(g) \in \mathcal{D}$ which is the degenerated hyper-rectangle $\delta(g) = \{\mathcal{M}(g)\} = \bigtimes_{i=1}^{p}[m_i(g), m_i(g)]$. The cover relationship becomes $d \sqsubseteq \delta(g)$. The *extent* of a pattern is the set of objects supporting it. Formally, there is a function $ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\} = \{g \in \mathcal{G} \mid \mathcal{M}(g) \in d\}$ (where $\wp(\mathcal{G})$ denotes the set of all subsets of $\mathcal{G}$). Note that if $d_1 \sqsubseteq d_2$ then $ext(d_2) \subseteq ext(d_1)$. We define also the positive (resp. negative) extent as follows: $ext^+(d) = ext(d) \cap \mathcal{G}^+$ (resp. $ext^-(d) = ext(d) \cap \mathcal{G}^-$). With the mapping $\delta : \mathcal{G} \to \mathcal{D}$ and the complete lattice $(\mathcal{D}, \sqsubseteq)$, we call the triple $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ the *interval pattern structure* [90, 62].

Measuring the discriminative power of a pattern. In SD, a quality measure $\phi : \mathcal{D} \to \mathbb{R}$ is usually defined to evaluate at what extent a pattern *well-discriminates* the positive instances in $\mathcal{G}^+$ from those in $\mathcal{G}^-$. Two atomic measures are generally employed to quantify the quality of a pattern d: the *true positive rate* $tpr : d \to |ext^+(d)|/|\mathcal{G}^+|$ and the *false positive rate* $fpr : d \to |ext^-(d)|/|\mathcal{G}^-|$. Several measures exist in the literature [68, 106]. A measure is said to be *objective* or *probability based* [68] if it depends solely on the number of co-occurrences and non co-occurrences of the pattern and the target label. In other words, those measures can be defined using only tpr, fpr and potentially other constants (e.g. $|\mathcal{G}|$). Formally, $\exists \phi^* : [0,1]^2 \to \mathbb{R}$ s.t. $\phi(d) = \phi^*(tpr(d), fpr(d))$. Objective measures depends only on the pattern *extent*. Hence, we use interchangeably $\phi(ext(d))$ and $\phi(d)$. An objective quality measure ϕ is said to be *discriminant* if its associated measure ϕ^* is *increasing* with tpr (fpr being fixed) and *decreasing* with fpr (tpr being fixed). For instance, with $\alpha^+ = |\mathcal{G}^+|/|\mathcal{G}|$ and $\alpha^- = |\mathcal{G}^-|/|\mathcal{G}|$ denoting labels prevalence, $wracc^*(tpr, fpr) = \alpha^+ \cdot \alpha^- \cdot (tpr - fpr)$ and $informedness^*(tpr, fpr) = tpr - fpr$ are discriminant measures.

Compressing the set of interesting patterns using closure. Since discriminant quality measures depend only on the extent, *closed patterns* can be leveraged to reduce the number of resulting patterns [62]. A pattern $d \in \mathcal{D}$ is said to be closed (w.r.t. pattern structure $\mathbb{P}$) if and only if it is the most restrictive pattern (i.e. the smallest hyper-rectangle) enclosing its extent. Formally, $d = int(ext(d))$ where int mapping (called *intent)* is given by: $int : \wp(\mathcal{G}) \to \mathcal{D}, A \mapsto \bigsqcap_{g\in A} \delta(g) = \bigtimes_{i=1}^{p}[\min_{g\in A} m_i(g), \max_{g\in A} m_i(g)]$. Fig. 5.1 (center-left) depicts the closed interval pattern (hatched rectangle) $c_1 = [1,2]\times[1,3]$ which is the closure of $c_2 = [1,4]\times[0,3]$ (non hatched rectangle). Note that since $\mathcal{G}$ is finite, the set of all closed patterns is finite and is given by $int[\wp(\mathcal{G})]$.

A more concise set of patterns using Relevance theory. Fig. 5.1 (center-right) depicts two interval patterns, the hatched pattern $d_1 = [1,2]\times[1,3]$ and the non-hatched one $d_2 = [1,4]\times[1,4]$. While both patterns are *closed*, d_1 has better discriminative power than d_2 since they both cover exactly the same positive instances $\{g_1, g_2, g_3\}$; yet, d_2 covers more negative instances than d_1. *Relevance theory* [66] formalizes this observation and helps us to remove some clearly uninteresting closed patterns. In a nutshell, a closed pattern $d_1 \in \mathcal{D}$ is said to be *more relevant than* a closed pattern $d_2 \in \mathcal{D}$ iff $ext^+(d_2) \subseteq ext^+(d_1)$ and $ext^-(d_1) \subseteq ext^-(d_2)$. For ϕ discriminant, if d_1 is more relevant than d_2 then $\phi(d_1) \geq \phi(d_2)$. A closed pattern d is said to be *relevant* iff there is no other closed pattern c that is more relevant than d.

	m_1	m_2	label
g_1	1	2	+
g_2	1	3	+
g_3	2	1	+
g_4	3	5	+
g_5	2	4	−
g_6	2	5	−
g_7	3	4	−
g_8	4	4	−

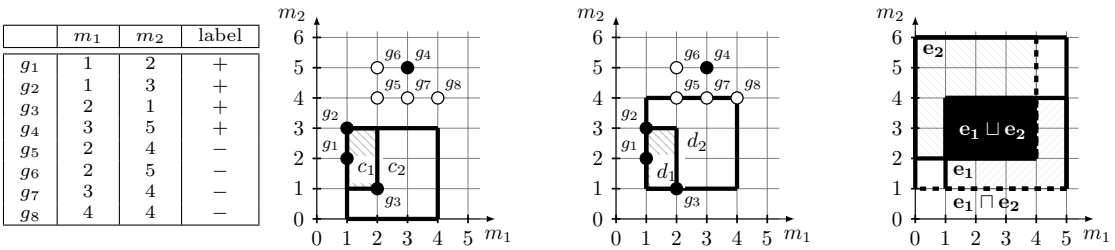


Figure 5.1: (left to right) (1) a labeled numerical dataset. **(2)** closed c_1 vs non-closed c_2 interval patterns. **(3)** `cotp` d_1 vs non `cotp` d_2. **(4)** meet and join of two patterns.

It follows that if a closed pattern is relevant then it is *closed on the positive* (`cotp` for short). An interval pattern is said to be `cotp` if any smaller interval pattern will at least drop one positive instance (i.e. $d = int(ext^+(d))$). interestingly, $int \circ ext^+$ is a closure operator on $(\mathcal{D}, \sqsubseteq)$. Fig. 5.1 (center-right) depicts a non `cotp` pattern $d_2 = [1,4] \times [1,4]$ and its closure on the positive $d_1 = int(ext^+(d_2)) = [1,2] \times [1,3]$ which is relevant. Note that not all `cotp` are *relevant*. The set of `cotp` patterns is given by $int[\wp(\mathcal{G}^+)]$. We call *relevant (resp.* `cotp`*) extent*, any set $A \subseteq \mathcal{G}$ s.t. $A = ext(d)$ with d is a *relevant (resp.* `cotp`*)* pattern. The set of relevant extents is denoted by $\mathcal{R}$.

5.3 Problem Statement

Correct enumeration of relevant extents. First, consider the (simpler) problem of enumerating all relevant extents in $\mathcal{R}$. For a (relevant extents) enumeration algorithm, three properties need generally to hold. An algorithm which output is the set of solutions $\mathcal{S}$ is said to be (1) *complete* if $\mathcal{S} \supseteq \mathcal{R}$, (2) *sound* if $\mathcal{S} \subseteq \mathcal{R}$ and (3) *non redundant* if each solution in $\mathcal{S}$ is outputted *only once*. It is said to be *correct* if the three properties hold. Guyet et al. [75] proposed a *correct algorithm* that enumerate relevant extents induced by the *interval pattern structure* in two steps: (1) Start by a *DFS complete* and *non redundant* enumeration of all `cotp` patterns (extents) using `MinIntChange` algorithm [90]; (2) Post-process the found `cotp` patterns by removing non relevant ones using [66] characterization (this step adds the *soundness* property to the algorithm).

Problem Statement. Given a discriminant objective quality measure ϕ, we want to design an *anytime enumeration algorithm* such that: (1) given enough time, outputs all relevant extents in $\mathcal{R}$, (2) when interrupted, provides a guarantee bounding the difference of quality between the top-quality found extent and the top possible quality w.r.t. ϕ; and (3) outputs a second guarantee ensuring that the resulting patterns are diverse.

Formally, let $\mathcal{S}_i$ be the set of outputted solutions by the anytime algorithm at some step (or instant) i (at $i+1$ we have $\mathcal{S}_i \subseteq \mathcal{S}_{i+1}$). We want that (1) when i is big enough, $\mathcal{S}_i \supseteq \mathcal{R}$ (only *completeness* is required). For (2) and (3), we define two metrics[8] to compare the results in $\mathcal{S}_i$ with the ones in $\mathcal{R}$. The first metric, called *accuracy* (eq. 5.1), evaluates the difference between top pattern quality ϕ in $\mathcal{S}_i$ and $\mathcal{R}$ while the second metric, called $specificity$ (eq. 5.2), evaluates how diverse and complete are patterns in $\mathcal{S}_i$.

$$accuracy_\phi(S_i, \mathcal{R}) = \sup_{A \in \mathcal{R}} \phi(A) - \sup_{B \in \mathcal{S}_i} \phi(B) \tag{5.1}$$

$$specificity(S_i, \mathcal{R}) = \sup_{A \in \mathcal{R}} \inf_{B \in \mathcal{S}_i} (|A \,\Delta\, B|/|\mathcal{G}|) \tag{5.2}$$

The idea behind $specificity$ is that each extent A in $\mathcal{R}$ is "approximated" by the most similar extent in $\mathcal{S}_i$; that is the set $B \in \mathcal{S}_i$ minimizing the *metric distance* $A, B \mapsto |A \,\Delta\, B|/|\mathcal{G}|$ in $\wp(\mathcal{G})$. The $specificity$[9] is then the highest possible distance (pessimistic). Note that $specificity(\mathcal{S}_i, \mathcal{R}) = 0$ *is equivalent to* $\mathcal{S}_i \supseteq \mathcal{R}$. Clearly, the lower these two metrics are, the closer we get to the desired output $\mathcal{R}$. While $accuracy_\phi$ and $specificity$ can be *evaluated* when a complete exploration of $\mathcal{R}$ is possible, our aim is to *bound* the two aforementioned measures independently from $\mathcal{R}$ providing a *guarantee*. In other words, the anytime algorithm need to output additionally to S_i, the two following measures: (2) $\overline{accuracy_\phi}(\mathcal{S}_i)$ and (3) $\overline{specificity}(\mathcal{S}_i)$ s.t. $accuracy_\phi(\mathcal{S}_i, \mathcal{R}) \leq \overline{accuracy_\phi}(\mathcal{S}_i)$ and $specificity(\mathcal{S}_i, \mathcal{R}) \leq \overline{specificity}(\mathcal{S}_i)$. These two bounds need to decrease overtime providing better information on $\mathcal{R}$ through $\mathcal{S}_i$.

5.4 Anytime Interval Pattern Mining

Discretizations and pattern space. Our algorithm relies on the enumeration of a chain of discretization from the coarsest to the finest. A *discretization* of $\mathbb{R}$ is any *partition* of $\mathbb{R}$ using intervals. In

[8] The metrics names fall under the taxonomy of [174] for anytime algorithms.

[9] The $specificity$ is actually a directed Hausdorff distance [81] from $\mathcal{R}$ to $\mathcal{S}_i$.

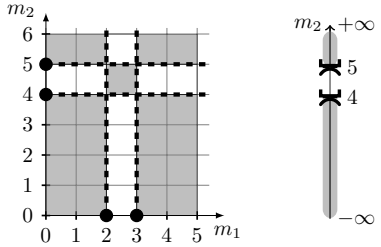


Figure 5.2: (left) Discretization $dr((C_1, C_2))$ in $\mathbb{R}^2$ with $C_1 = \{2, 3\}$ and $C_2 = \{4, 5\}$ and **(right)** discretization $dr((C_2))$ in $\mathbb{R}$. Adding a cut point in any C_k will create finer discretization.

particular, let $C = \{c_i\}_{1 \leq i \leq |C|} \subseteq \mathbb{R}$ be a finite set with $c_i < c_{i+1}$ for $i \in \{1, ..., |C| - 1\}$. Element of C are called *cut points* or *cuts*. We associate to C a *finite discretization* denoted by $dr(C)$ and given by $dr(C) = \{(-\infty, c_1)\} \cup \{[c_i, c_{i+1}) \mid i \in \{1, ..., |C| - 1]\}\} \cup \big\{[c_{|C|}, +\infty)\big\}$.

Generally speaking, let $p \in \mathbb{N}^*$ and let $C = (C_k)_{1 \leq k \leq p} \in \wp(\mathbb{R})^p$ representing *sets of cut points* associated to each dimension k (i.e. $C_k \subseteq \mathbb{R}$ finite $\forall k \in \{1, ..., p\}$). The partition $dr(C)$ of $\mathbb{R}^p$ is given by: $dr(C) = \prod_{k=1}^{p} dr(C_k)$. Fig. 5.2 depicts two discretizations. Discretizations are ordered using the natural order between partitions[10]. Moreover, cut-points sets are ordered by $\leq$ as follows: $C^1 \leq C^2 \equiv (\forall k \in \{1, ..., p\})\ C_k^1 \subseteq C_k^2$ with $C^i = (C_k^i)_{1 \leq k \leq p}$. Clearly, if $C^1 \leq C^2$ then discretization $dr(C^1)$ *is coarser than* $dr(C^2)$.

Let $C = (C_k)_{1 \leq k \leq p}$ be the cut-points. Using the elementary hyper-rectangles (i.e. cells) in the discretization $dr(C)$, one can build a (finite) subset of descriptions $\mathcal{D}_C \subseteq \mathcal{D}$ which is the set of all possible descriptions (hyper-rectangles) that can be built using these cells. Formally: $\mathcal{D}_C = \{\bigsqcap S \mid S \subseteq dr(C)\}$. Note that $\top = \emptyset \in \mathcal{D}_C$ since $\bigsqcap \emptyset = \bigsqcup \mathcal{D} = \top$ by definition. Proposition 13 states that $(\mathcal{D}_C, \sqsubseteq)$ is a complete sub-lattice of $(\mathcal{D}, \sqsubseteq)$.

Proposition 13 *$(\mathcal{D}_C, \sqsubseteq)$ is a finite (complete) sub-lattice of $(\mathcal{D}, \sqsubseteq)$ that is: $\forall d_1, d_2 \in \mathcal{D}_C : d_1 \sqcup d_2 \in \mathcal{D}_C$ and $d_1 \sqcap d_2 \in \mathcal{D}_C$. Moreover, if $C^1 \leq C^2$ are two cut-points sets, then $(\mathcal{D}_{C^1}, \sqsubseteq)$ is a (complete) sub-lattice of $(\mathcal{D}_{C^2}, \sqsubseteq)$.*

Finest discretization for a complete enumeration of relevant extents. There exist cut points $C \subseteq \wp(\mathbb{R})^p$ such that the space $(\mathcal{D}_C, \sqsubseteq)$ holds all relevant extents (i.e. $ext[\mathcal{D}_C] \supseteq \mathcal{R}$). For instance, if we consider $C = (m_k[\mathcal{G}])_{1 \leq k \leq p}$, the description space $(\mathcal{D}_C, \sqsubseteq)$ holds all relevant extents. However, is there coarser discretization that holds all the relevant extents? The answer is affirmative. One can show that the only interesting cuts are those separating between positive and negative instances (called boundary cut-points by [57]). We call such cuts, *relevant cuts*. They are denoted by $C^{rel} = (C_k^{rel})_{1 \leq k \leq p}$ and we have $ext[\mathcal{D}_{C^{rel}}] \supseteq \mathcal{R}$. Formally, for each dimension k, a value $c \in m_k[\mathcal{G}]$ is a *relevant cut in* C_k^{rel} for attribute m_k iff: $(c \in m_k[\mathcal{G}^+]$ and $prev(c, m_k[\mathcal{G}]) \in m_k[\mathcal{G}^-])$ or $(c \in m_k[\mathcal{G}^-]$ and $prev(c, m_k[\mathcal{G}]) \in m_k[\mathcal{G}^+])$ where $next(c, A) = \inf\{a \in A \mid c < a\}$ (resp. $prev(c, A) = \sup\{a \in A \mid a < c\}$) is the following (resp. preceding) element of c in A. Finding *relevant cuts* C_k^{rel} is of the same complexity of sorting $m_k[\mathcal{G}]$ [57]. In the dataset depicted in Fig. 5.1, relevant cuts are given by $C^{rel} = (\{2, 3, 4, 5\}, \{4, 5\})$. Discretization $dr(C_2^{rel})$ is depicted in Fig. 5.2 (center).

[10] Let E be a set, a partition P_2 of E *is finer than* a partition P_1 (or P_1 *is coarser than* P_2) and we denote $P_1 \leq P_2$ if any subset in P_1 is a subset of a subset in P_2.

Anytime enumeration of relevant extents. We design an *anytime* and *interruptible* algorithm dubbed `RefineAndMine`. This method, presented in Algorithm 7, relies on the enumeration of a chain of discretizations on the data space, from the coarsest to the finest. It begins by searching relevant cuts in pre-processing phase (line 2). Then, it builds a *coarse discretization* (line 3) containing a small set of relevant cut-points. Once the initial discretization built, `cotp` patterns are mined thanks to `MinIntChange` Algorithm (line 4) [90]. Then as long as the algorithm is not interrupted (or within the computational budget), we add new cut-points (line 6) building finer discretizations. For each added cut-point (line 8), only new interval patterns are searched for (mined descriptions d are new but their extents $ext(d)$ are not necessarily new) . That is `cotp` patterns which left or right bound is *cut* on the considered attribute *attr* (i.e. $d.I_{attr} \in \{[cut, a), [cut, +\infty), [a, cut), (-\infty, cut) \mid a \in C^{cur}_{attr}\}$ with $d.I_{attr}$ is the $attr^{th}$ interval of d). This can be done by a slight modification of `MinIntChange` method. `RefineAndMine` terminates when the set of relevant cuts is exhausted (i.e. $C^{cur} = C^{rel}$) ensuring a *complete* enumeration of relevant extents $\mathcal{R}$.

The initial discretization (Line 3) can be done by various strategies (see [171]). A simple, yet efficient, choice is the equal frequency discretization with a fixed number of cuts. Other strategies can be used, e.g. [57]. Adding new cut-points (Line 6) can also be done in various ways. One strategy is to add a random relevant cut on a random attribute to build the next discretization. Section 5.5.3 proposes another more elaborated strategy that heuristically guide `RefineAndMine` to rapidly find good quality patterns (observed experimentally).

Algorithm 7 `RefineAndMine`

1: Input: $(\mathcal{G}, \mathcal{M})$ a numerical datasets with $\{\mathcal{G}^+, \mathcal{G}^-\}$ partition of $\mathcal{G}$
2: Compute relevant cuts C^{rel}
3: Build an initial set of cut-points $C^{cur} \leq C^{rel}$
4: Mine `cotp` patterns in $\mathcal{D}_{C^{cur}}$ (and their extents) using `MinIntChange`
5: **while** $C^{cur} \neq C^{rel}$ and within **computational budget do**
6: Choose the next relevant cut $(attr, cut)$ with $cut \in C^{rel}_{attr} \backslash C^{cur}_{attr}$
7: Add the relevant cut *cut* to C^{cur}
8: Mine **new `cotp` patterns** (and their extents) in $\mathcal{D}_{C^{cur}}$
9: **end while**

5.5 Anytime Interval Pattern Mining with Guarantees

Algorithm `RefineAndMine` starts by mining patterns in a coarse discretization. It continues by mining more patterns in increasingly finer discretizations until the search space is totally explored (final complete lattice being $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$). According to Proposition 13, the description spaces built on discretizations are complete sub-lattices of the total description space. A similar idea involves performing successive enumeration of growing pattern languages (projections) [37]. In our case, it is a successive enumeration of growing complete sub-lattices. For the sake of generality, in the following of this section $(\mathcal{D}, \sqsubseteq)$ denotes a *complete lattice*, and for all $i \in \mathbb{N}^*$, $(\mathcal{D}_i, \sqsubseteq)$ denotes *complete sub-lattices* of $(\mathcal{D}, \sqsubseteq)$ such that $\mathcal{D}_i \subseteq \mathcal{D}_{i+1} \subseteq \mathcal{D}$. For instance, in `RefineAndMine`, the total complete lattice is $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$ while the $(\mathcal{D}_i, \sqsubseteq)$ are $(\mathcal{D}_{C^{cur}}, \sqsubseteq)$ at each step. Following Sec. 5.3 notation, the outputted set $\mathcal{S}_i$ at a step i contains the set of all ***cotp*** *extents* associated to $\mathcal{D}_i$. Before giving the formulas of $\overline{accuracy_\phi}(\mathcal{S}_i)$ and $\overline{specificity}(\mathcal{S}_i)$, we give some necessary definitions and underlying properties. At the end of this section, we show how `RefineAndMine` can be adapted to efficiently compute these two bounds for the case of interval patterns.

Similarly to the interval pattern structure [90], we define in the general case a *pattern structure* $\mathbb{P} = (\mathcal{G}, (\mathcal{D}, \sqsubseteq), \delta)$ on the *complete lattice* $(\mathcal{D}, \sqsubseteq)$ where $\mathcal{G}$ is a non empty finite set (partitioned into $\{\mathcal{G}^+, \mathcal{G}^-\}$) and $\delta : \mathcal{G} \to \mathcal{D}$ is a mapping associating to each object its description (recall that in interval pattern structure, δ is the degenerated hyper-rectangle representing a single point). The extent *ext* and intent *int* operators are then respectively given by $ext : \mathcal{D} \to \wp(\mathcal{G}), d \mapsto \{g \in \mathcal{G} \mid d \sqsubseteq \delta(g)\}$ and $int : \wp(\mathcal{G}) \to \wp(\mathcal{G}), A \mapsto \bigsqcap_{g \in A} \delta(g)$ with $\bigsqcap$ represents the meet operator in $(\mathcal{D}, \sqsubseteq)$ [62].

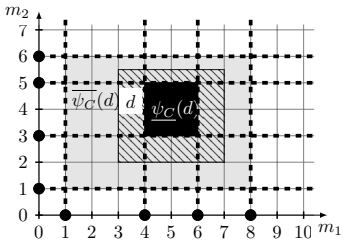


Figure 5.3: Description $d = [3,7] \times [2,5.5]$ in $\mathcal{D}$ (hatched) and $C = (\{1,4,6,8\},\{1,3,5,6\})$. Upper approximation of d in $\mathcal{D}_C$ is $\overline{\psi_C}(d) = [1,8) \times [1,6)$ (gray rectangle) while lower approximation of d is $\underline{\psi_C}(d) = [4,6) \times [3,5)$ (black rectangle).

5.5.1 Approximating descriptions in a complete sub-lattice

Upper and lower approximations of a pattern. We start by approximating each pattern in $\mathcal{D}$ using two patterns in $\mathcal{D}_i$. Consider for instance Fig. 5.3 where $\mathcal{D}$ is the space of interval patterns in $\mathbb{R}^2$ while $\mathcal{D}_C$ is the space containing only rectangles that can be built over discretization $dr(C)$ with $C = (\{1,4,6,8\},\{1,3,5,6\})$. Since the hatched rectangle $d = [3,7] \times [2,5.5] \in \mathcal{D}$ does not belong to $\mathcal{D}_C$, two descriptions in $\mathcal{D}_C$ can be used to encapsulate it. The first one, depicted by a gray rectangle, is called the *upper approximation* of d. It is given by the smallest rectangle in $\mathcal{D}_C$ enclosing d. Dually, the second approximation represented as a black rectangle and coined *lower approximation* of d, is given by the greatest rectangle in $\mathcal{D}_C$ enclosed by d. This two denominations comes from Rough Set Theory [131] where lower and upper approximations form together a *rough set* and try to capture the undefined rectangle $d \in \mathcal{D} \backslash \mathcal{D}_C$. Definition 41 formalizes these two approximations in the general case.

Definition 41 *The upper approximation mapping $\overline{\psi_i}$ and lower approximation mapping $\underline{\psi_i}$ are the mappings defined as follows:*

$$\overline{\psi_i} : \mathcal{D} \to \mathcal{D}_i, d \mapsto \bigsqcup \{c \in \mathcal{D}_i \mid c \sqsubseteq d\} \qquad \underline{\psi_i} : \mathcal{D} \to \mathcal{D}_i, d \mapsto \bigsqcap \{c \in \mathcal{D}_i \mid d \sqsubseteq c\}$$

The existence of these two mappings is ensured by the fact that $(\mathcal{D}_i, \sqsubseteq)$ is a complete sublattice of $(\mathcal{D}, \sqsubseteq)$. *Theorem 4.1* in [49] provides more properties for the two aforementioned mappings. Proposition 14 restates an important property.

Proposition 14 $\forall d \in \mathcal{D} : \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d)$. *The term lower and upper-approximation here are reversed to fit the fact that in term of* extent *we have* $\forall d \in \mathcal{D}$: $ext(\underline{\psi_i}(d)) \subseteq ext(d) \subseteq ext(\overline{\psi_i}(d))$.

A projected pattern structure. Now that we have the upper-approximation mapping $\overline{\psi_i}$, one can associate a new pattern structure $\mathbb{P}_i = (\mathcal{G}, (\mathcal{D}_i, \sqsubseteq), \overline{\psi_i} \circ \delta)$[11] to the pattern space $(\mathcal{D}_i, \sqsubseteq)$. It is worth mentioning, that while extent ext_i mapping associated to $\mathbb{P}_i$ is equal to ext, the intent int_i of $\mathbb{P}_i$ is given by $int_i : \wp(\mathcal{G}) \to \mathcal{D}_i, A \mapsto \overline{\psi_i}(int(A))$. Note that, the set of `cotp` patterns associated to $\mathbb{P}_i$ are given by $int_i[\wp(\mathcal{G}^+)] = \overline{\psi_i}[int[\wp(\mathcal{G}^+)]]$. That is, the upper approximation of a `cotp` pattern in $\mathbb{P}$ is a `cotp` pattern in $\mathbb{P}_i$.

Encapsulating patterns using their upper-approximations. We want to encapsulate any description by knowing only its upper-approximation. Formally, we want some function $f : \mathcal{D}_i \to \mathcal{D}_i$ such that $(\forall d \in \mathcal{D})\, \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq f(\overline{\psi_i}(d))$. Proposition 15 define such a function f (called *core*) and states that the *core* is the tightest (w.r.t. $\sqsubseteq$) possible function f.

Proposition 15 *The function $core_i$ defined by:*

$$core_i : \mathcal{D}_i \to \mathcal{D}_i, c \mapsto core(c) = \underline{\psi_i}\left(\bigsqcup \left\{d \in \mathcal{D} \mid \overline{\psi_i}(d) = c\right\}\right)$$

verifies the following property: $\forall d \in \mathcal{D} : \overline{\psi_i}(d) \sqsubseteq d \sqsubseteq \underline{\psi_i}(d) \sqsubseteq core_i(\overline{\psi_i}(d))$. *Moreover, for* $f : \mathcal{D}_i \to \mathcal{D}_i$, $(\forall d \in \mathcal{D})\, d \sqsubseteq f(\overline{\psi_i}(d)) \Leftrightarrow (\forall c \in \mathcal{D}_i)\, core_i(c) \sqsubseteq f(c)$.

[11] $\mathbb{P}_i$ is said to be a projected pattern structure of $\mathbb{P}$ by the projection $\overline{\psi_i}$ [38].

Note that, while the *core* operator definition depends clearly on the complete lattice $(\mathcal{D}, \sqsubseteq)$, its computation should be done independently from $(\mathcal{D}, \sqsubseteq)$.

We show here how to compute the *core* in `RefineAndMine`. In each step and for cut-points $C = (C_k) \subseteq \wp(\mathbb{R})^p$, the finite lattice $(\mathcal{D}_C, \sqsubseteq)$ is a sub-lattice of the finest finite lattice $(\mathcal{D}_{C^{rel}}, \sqsubseteq)$ (since $C \leq C^{rel}$). Thereby, the *core* is computed according to this latter as follows: Let $d \in \mathcal{D}_C$ with $d.I_k = [a_k, b_k)$ for all $k \in \{1, ..., p\}$. The left (resp. right) bound of $core_C(d).I_k$ for any k is equal to $next(a_k, C_k)$ (resp. $prev(b_k, C_k)$) if $next(a_k, C_k^{rel}) \notin C_k$ (resp. $prev(b_k, C_k^{rel}) \notin C_k$). Otherwise, it is equal to a_k (resp. b_k). Consider the step $\mathcal{C} = (\{2, 3\}, \{4, 5\})$ in `RefineAndMine` (its associated discretization is depicted in Fig. 5.2 (left)) and recall that the relevant cuts set is $\mathcal{C}^{rel} = (\{2, 3, 4, 5\}, \{4, 5\})$. The core of the bottom pattern $\bot = \mathbb{R}^2$ at this step is $core_{C^{cur}}(\bot) = (-\infty, 3) \times \mathbb{R}$. Indeed, there is three descriptions in $\mathcal{D}_{C^{rel}}$ which upper approximation is $\bot$, namely $\bot$, $c_1 = (-\infty, 4) \times \mathbb{R}$ and $c_2 = (-\infty, 5) \times \mathbb{R}$. Their lower approximations are respectively $\bot$, $(-\infty, 3) \times \mathbb{R}$ and $(-\infty, 3) \times \mathbb{R}$. The join (intersection) of these three descriptions is then $core_{C^{cur}}(\bot) = (-\infty, 3) \times (-\infty, +\infty)$. Note that particularly for interval patterns, the *core* has *monotonicity*, that is $(\forall c, d \in \mathcal{D}_C)\, c \sqsubseteq d \Rightarrow core_C(c) \sqsubseteq core_C(d)$.

5.5.2 Bounding accuracy and specificity metrics

At the i^{th} step, the outputted extents $\mathcal{S}_i$ contains the set of `cotp` extents in $\mathbb{P}_i$. Formally, $int_i[S_i] \supseteq int_i[\wp(\mathcal{G}^+)]$. Theorem 2 and Theorem 3 gives respectively the *bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$.

Theorem 2 *Let $\phi : \mathcal{D} \to \mathbb{R}$ be a discriminant objective quality measure. The accuracy metric is bounded by:*

$$\overline{accuracy_\phi}(\mathcal{S}_i) \quad = \quad \sup_{c \in int_i[\mathcal{S}_i]} \left[\phi^* \left(tpr\big(c\big), fpr\big(core_i(c)\big)\right) - \phi^* \left(tpr(c), fpr(c)\right)\right]$$

Moreover $\overline{accuracy_\phi}(\mathcal{S}_{i+1}) \leq \overline{accuracy_\phi}(\mathcal{S}_i)$.

Theorem 3 *The $specificity$ metric is bounded by:*

$$\overline{specificity}(\mathcal{S}_i) \quad = \quad \sup_{c \in int_i[\mathcal{S}_i]} \Big(\big(|ext(c)| - |ext(core_i^+(c))|\big)/(2 \cdot |\mathcal{G}|)\Big)$$

where $core_i^+(c) = int_i(ext^+(core_i(c)))$, that is $core_i^+(c)$ is the closure on the positive of $core_i(c)$ in $\mathbb{P}_i$. Moreover $\overline{specificity}(\mathcal{S}_{i+1}) \leq \overline{specificity}(\mathcal{S}_i)$.

5.5.3 Computing and updating bounds in `RefineAndMine`

We show below how the different steps of the method `RefineAndMine` (see Algorithm 7) should be updated in order to compute the two bounds $\overline{accuracy}$ and $\overline{specificity}$. For the sake of brevity, we explain here a naive approach to provide an overview of the algorithm. Note that here, *core* (resp. $core^+$) refers to $core_{C^{cur}}$ (resp. $core^+_{C^{cur}}$).

Compute the initial bounds (line 4). As `MinIntChange` enumerates all `cotp` patterns $d \in \mathcal{D}_{C^{cur}}$, `RefineAndMine` stores in a key-value structure (i.e. map) called `BoundPerPosExt` the following entries:

$$ext^+(d) : \left(\phi(d), \phi^*\left(tpr\big(d\big), fpr\big(core(d)\big)\right), \big(|ext(d)| - |ext(core^+(d))|\big)/(2 \cdot |\mathcal{G}|)\right)$$

The *error-bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$ are then computed at the end by a single pass on the entries of `BoundPerPosExt` using Theorems 2 and 3.

Update the bounds after adding a new cut-point (line 8). In order to compute the new *error-bounds* $\overline{accuracy_\phi}$ and $\overline{specificity}$ which decrease according to theorems 2 and 3, one need to add/update some entries in the structure `BoundPerPosExt`. For that, only two types of patterns should be looked for:

1. The new `cotp` patterns mined by `RefineAndMine`, that is those which left or right bound on attribute *attr* is the added value *cut*. Visiting these patterns will add potentially new entries in `BoundPerPosExt` or update ancient ones.
2. The old `cotp` which core changes (i.e. becomes less restrictive) in the new discretization. One can show that these patterns are those which left bound is $prev(cut, C^{cur}_{attr})$ or right bound is $next(cut, C^{cur}_{attr})$ on attribute *attr*. Visiting these patterns will only update ancient entries of `BoundPerPosExt` by potentially decreasing both second and third value.

Adding a new cut-point (line 7). We have implemented for now a strategy which aims to decrease the $\overline{accuracy_\phi}$. For that, we search in `BoundPerPosExt` for the description d having the maximal value $\phi^*\big(tpr\big(d\big), fpr\big(core(d)\big)\big)$. In order to decrease $\overline{accuracy_\phi}$, we increase the size of $core(d)$ (to potentially increase $fpr\big(core(d)\big)$). This is equivalent to choose a cut-point in the border region $C^{rel}_{attr} \backslash C^{cur}_{attr}$ for some attribute *attr* such that $cut \in d.I_{attr} \backslash core(d).I_{attr}$. Consider that we are in the step where the current discretization C^{cur} is the one depicted in Fig. 5.2. Imagine that the bottom pattern $\bot = \mathbb{R}^2$ is the one associated to the maximal value $\phi^*\big(tpr\big(\bot\big), fpr\big(core(\bot)\big)\big)$. The new cut-point should be chosen in $\{4, 5\}$ for $attr = 1$ (recall that $core(\bot) = (-\infty, 3) \times (-\infty, +\infty)$). Note that if for such description there is no remaining relevant cut in its *border regions* for all $attr \in \{1, ..., p\}$ then $core(d) = d$ ensuring that d is the *top pattern*.

5.6 Conclusion

We introduced a novel anytime pattern mining technique for uncovering discriminant patterns in numerical data. We took a close look to discriminant interestingness measures to focus on hyper-rectangles in the dataset fostering the presence of some class. By leveraging the properties of the quality measures, we defined a guarantee on the accuracy of `RefineAndMine` in approximating the optimal solution which improves over time. We also presented a guarantee on the specificity of `RefineAndMine` –which is agnostic of the quality measure– ensuring its diversity and completeness. An empirical evaluation (not reported in this manuscript but in the original article) gives evidence of the effectiveness both in terms of finding the optimal solution (w.r.t. the quality measure ϕ) and revealing local optimas located in different parts of the data.

This work paves the way for many improvements. `RefineAndMine` can be initialized with more sophisticated discretization techniques [96, 57]. We have to investigate additional cut-points selection strategies. While we considered here discriminant pattern mining, the enumeration process (i.e. *successive refinement of discretizations*) can be tailored to various other quality measures in subgroup discovery. For example, the accuracy bound guarantee definition can be extended to handle several other traditional measures such as Mutual Information, χ^2 and *Gini split* by exploiting their (quasi)-convexity properties w.r.t. *tpr* and *fpr* variables [124, 3]. Other improvements include the adaptation of `RefineAndMine` for high-dimensional datasets and its generalization for handling additional types of attributes (categorical, itemsets, etc.). The latter is facilitated by the generic notions from Section 5.5 and the recent works of Buzmakov et al. [37].

Part III

Video Game Analytics

Chapter 6

Discovering Opening Strategies in RTS Games

6.1 Introduction

The recent and fast development of the video game industry has been catalyzed by technological innovations, a democratized access to connected electronic devices, new economic models (*free* games where users may pay for extra contents), and recently with competitive gaming (esports) and video game live streaming platforms [152]. People not only enjoy playing, but also enjoy learning from watching others performing, as a daily leisure activity [45, 93], as we remarked an early adoption of Twitch.tv in 2011 which today in 2018 followed in greater numbers.

Producing a video game is an expensive process, thus, keys to a massive, immediate and durable success are sought. Pragmatically however, one attempts to extend the game lifetime after the release, by correcting bugs, introducing new features and considering user feedbacks. Whereas it could be easily argued that bugs are not acceptable after a release, it is hard to predict the results of human creativity in presence of rich environments that are video games.

Hopefully, companies realized in the current *big data* atmosphere that the tremendous amounts of game behavioral data they store are valuable to face many new challenges such as: detecting unexpected situations and bugs [161] cheaters [10], designing artificial agents [128], improving match making systems and adjusting game difficulty [120]. Analyzing these massive sets of historical data by means of visualization, machine learning and data mining techniques is at the heart of *video game analytics* for enhancing user experience and extending game lifetime [161].

This context roots the motivation of our work: behavioral data can help to study the balance of a game, that is, to adjust the rules over time while still enabling novel rules to counter boredom. This is especially important for games played as an *electronic sport*, and also for game lifetime extension in general. We will focus on the concept of *balance* which is a core concept in competitive game design: it consists of defining and tuning the basic rules that prevent extreme situations, thus balancing fairness and competitive aspects.

In this article, we define and mine *patterns* in game historical data for a better understanding of balance issues in RTS games. The intuition of the *balanced sequential pattern discovery* problem is the following. Consider a set of games, each of them represented by a sequence of actions of two players (thus entailing the player strategy). The problem is to find patterns as sequence generalizations that frequently occur in the historical data and whose balance is given by proportions of their wins and losses. Our goal is twofold: (i) we give the basic algorithmic tools that enable an efficient pattern mining, (ii) we show that the extracted patterns reveal interesting knowledge:

- (i) We revisit the problem of strategy elicitation from two player RTS games by differentiating two

cases: when both players have access to (a) different game actions and (b) the same game actions. In the first case, we show how existing pattern mining methods enable with slight modifications to discover frequent strategies and compute a balance measure. In the second case, the most general, existing approaches fail: we propose an original algorithm, BalanceSpan.

- (ii) We show through experiments that BalanceSpan is scalable and able to discover patterns of interest in a large StarCraft II dataset that can help detecting balance issues. For that matter, we anchor our algorithm in a Knowledge Discovery in Databases process [59]. Pattern mining is one of the many steps of this interactive and iterative process guided by an expert of the data domain who selects and interprets the patterns [5, 69].

The chapter is organized as follows. Section 6.2 recalls the basics of sequential pattern mining before the introduction of our mining problem in Section 6.3. Our method is developed in sections 6.4 and 6.5. Algorithms are designed (Section 6.6) and experimented with E-Sport data (Section 6.7) before over-viewing related work and concluding.

6.2 Preliminaries

We recall the basic definitions of frequent sequential patterns [133] and emerging patterns [52] useful in the sequel. Let $\mathcal{I}$ be a finite set of *items*. Any non-empty subset $X \subseteq \mathcal{I}$ is called an *itemset*. A *sequence* $s = \langle X_1, ..., X_l \rangle$ is an ordered list of $l > 0$ itemsets. l is the length of the sequence, whereas $\sum_{i=1}^{l} |X_i|$ is its size. Considering $\mathcal{I}$ as a set of events (or actions), an itemset denotes simultaneous events while the order between two itemsets indicates a strict preceding relation. A sequence database $\mathcal{D}$ is a set of $|\mathcal{D}|$ sequences over $\mathcal{I}$. Sequences may have different lengths and sizes and are uniquely identified, see Table 6.1 (omitting the third column).

Definition (subsequence). A sequence $s = \langle X_1, ..., X_{l_s} \rangle$ is a *subsequence* of a sequence $s' = \langle X'_1, ..., X'_{l'_s} \rangle$, denoted $s \sqsubseteq s'$, if there exists $1 \leq j_1 < j_2 < ... < j_{l_s} \leq l'_s$ such that $X_1 \subseteq X'_{j_1}, X_2 \subseteq X'_{j_2}, ..., X_{l_s} \subseteq X'_{j_{l_s}}$.

Definition (Support and frequency) The *support* of a sequence s in a database $\mathcal{D}$ is $sup(s, \mathcal{D}) = \{s' \in \mathcal{D} \mid s \sqsubseteq s'\}$. Its frequency is $freq(s, \mathcal{D}) = |sup(s, \mathcal{D})|/|\mathcal{D}|$.

Problem (Frequent sequential pattern mining). Given a minimal frequency threshold $0 < \sigma \leq 1$, the problem is to find all sequences s such as $freq(s, \mathcal{D}) \geq \sigma$.

In some cases, each sequence of $\mathcal{D}$ is associated to a class label. Let $class : \mathcal{D} \to \{+, -\}$ a mapping that associates to each sequence a positive or negative label (hence two classes). $\mathcal{D}$ is accordingly partitioned into two databases, with the positive (resp. negative) sequences $\mathcal{D}^+$ (resp. $\mathcal{D}^-$) and $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$, $\mathcal{D}^+ \cap \mathcal{D}^- = \emptyset$. The growth-rate characterizes the discriminating power of a pattern for one class[52, 126].

Definition (Growth-rate). Given a sequence database $\mathcal{D} = \mathcal{D}^+ \cup \mathcal{D}^-$, the growth-rate of a sequential pattern from $\mathcal{D}^x$ to $\mathcal{D}^y$ ($x \neq y$ and $x, y \in \{+, -\}$), is given by

$$growth\_rate(s, \mathcal{D}^x, \mathcal{D}^y) = \frac{|sup(s, \mathcal{D}^x)|}{|\mathcal{D}^x|} \times \frac{|\mathcal{D}^y|}{|sup(s, \mathcal{D}^y)|}$$

Example. Let $\mathcal{D} = \{s_1, s_2, s_3, s_4\}$ with $\mathcal{I} = \{a,\ b,\ c,\ d,\ e,\ f,\ g\}$ be the sequence database given in Table 6.1. For brevity, we drop commas and braces for singletons. For a given sequence $s = \langle abc \rangle$, we have $s \sqsubseteq s_1$, $s \sqsubseteq s_4$, $s \not\sqsubseteq s_2$, $s \not\sqsubseteq s_3$. With $\sigma = \frac{3}{4}$, $\langle acc \rangle$ is frequent, $\langle a\{bc\}a \rangle$ is not. We have $growth\_rate(\langle cb \rangle, \mathcal{D}^-, \mathcal{D}^+) = \frac{2}{2} \times \frac{2}{1} = 2$, i.e., $\langle cb \rangle$ is twice more present in class $-$ than in class $+$.

id	$s \in \mathcal{D}$	$class(s)$
s_1	$\langle a\{abc\}\{ac\}d\{cf\}\rangle$	$+$
s_2	$\langle\{ad\}c\{bc\}\{ae\}\rangle$	$+$
s_3	$\langle\{ef\}\{ab\}\{df\}cb\rangle$	$-$
s_4	$\langle eg\{af\}cbc\rangle$	$-$

Table 6.1: A sequence database $\mathcal{D}$.

6.3 The Problem of Strategy Elicitation

A zero-sum game, or competitive interaction, can be modeled as a sequence of actions performed by two players where exactly one player wins (no ties). Such a sequential game can be represented as a sequence of sets of actions, each action performed by one of the two players. When both players play in real-time, one can describe these interactions as sequences of itemsets. An itemset is then a set of simultaneous actions, or within an insignificant interval of time.

Definition (Interaction (sequence) database). Given a set of players $Players$ and a set of actions $Actions$, a sequence database $\mathcal{R}$ is called an *interaction database*. Each sequence denotes one single game, i.e., an interaction between two players, and is defined over the set of items $\mathcal{I} = Actions \times Players$. A mapping $class : \mathcal{R} \rightarrow Players$ gives the winner of each interaction.

Example. In Table 6.2, s_1 can be interpreted as: "*Player p_1 did action a, then he did b and c while player p_2 did c, and finally p_1 did d while p_2 did a. At the end, the player p_1 wins*".

Given an interaction database, the problem is to find sequences of actions of both interacting players (supposing that those actions are mutually dependent) as generalizations that appear frequently and to be able to characterize their discriminating ability for a win or loss through a so-called *balance* measure. In the current sequential pattern mining settings, the goal is to find frequent sub-sequences of actions (i.e., strategies) and their balance (a growth-rate like measure). However, the notion of class has to be revisited to be able to handle winner and loser class labels, instead of the winning player. Indeed, intuitively, mining emerging patterns from an interaction database with the winning players as classes (as given in Table 6.2) does not fulfill our objectives: we wish to discriminate victories and not victorious players themselves. As such, existing emerging sequential pattern mining methods and algorithms cannot be used to answer our problem.

We propose to differentiate two cases of interaction databases: (i) *non-mirror interaction databases* where both players have different (non-intersecting) sets of available actions; (ii) *mirror interaction databases* where both players can perform the same actions. We show that in the first case, emerging patterns as introduced in the literature (Section II) are able to answer the problem by slightly modifying the interaction database representation. In the second case, the most general one, we need new settings, and we propose to embed the class (positive or negative) in the definition of the items of a sequence, see Table 6.4. This is formalized in the two next sections, and it enables the design of efficient pattern mining algorithms in Section 6.6.

id	Interaction sequence	Winner
s_1	$\langle(p_1,a)\{(p_1,b)(p_1,c)(p_2,c)\}\{(p_2,a)(p_1,d)\}\rangle$	p_1
s_2	$\langle(p_3,a)\{(p_3,b)(p_3,c)(p_3,d)\}\{(p_1,b)(p_1,c)\}\rangle$	p_3

Table 6.2: An interaction sequence database $\mathcal{R}$.

Table 6.3: A non mirror interaction database.

id	Interaction sequence	Winner
s_1	$\langle\{ab\}\{c\}\rangle$	$\mathcal{I}_1$
s_2	$\langle\{ab\}\{a\}\{c\}\rangle$	$\mathcal{I}_1$
s_3	$\langle\{abc\}\{c\}\rangle$	$\mathcal{I}_2$
s_4	$\langle\{ac\}\{a\}\{c\}\rangle$	$\mathcal{I}_1$
s_5	$\langle\{c\}\{b\}\{ac\}\rangle$	$\mathcal{I}_2$

6.4 Balanced Patterns in Non-mirror Databases

In this section, we consider an interaction database, called a *non-mirror database*, where the set of actions is different for each of the players in a single interaction. It means that we only have two types of players in each sequence and in the whole database (e.g., Protoss and Zerg factions in the RTS game StarCraft II), and these types are determined by the actions they can do. As such, the type can also be used as a winning class label. To characterize balanced patterns in such databases, we consider a simple transformation of the original interaction database $\mathcal{R}$ by dropping the player associated to each action, and labeling each sequence by the type of the winner. This enables to express a balance measure as a growth-rate measure in this new data representation. The transformed database is then formally defined as follows.

Definition (Transformed interaction database). A sequence database $\mathcal{T}$ defined over the set of items (actions) $\mathcal{I}_1 \cup \mathcal{I}_2$ such as $\mathcal{I}_1 \cap \mathcal{I}_2 = \emptyset$ and $class : \mathcal{T} \rightarrow \{\mathcal{I}_1, \mathcal{I}_2\}$ is called a *transformed interaction database*.

Consider an interaction $s \in \mathcal{T}$ where the winner is characterized by the actions $\mathcal{I}_1$: we have $class(s) = \mathcal{I}_1$ that gives the winner of the interaction. This brings back the problem of finding frequent balanced interaction patterns to well-known the emerging patterns settings. Indeed, consider an arbitrary pattern s over $\mathcal{I}_1 \cup \mathcal{I}_2$: its support in the whole database $sup(s,\mathcal{T})$ tells us its frequency, while $sup(s,\mathcal{T}^{\mathcal{I}_1})$ and $sup(s,\mathcal{T}^{\mathcal{I}_2})$ enable to define a balance measure as a growth-rate.

Problem (Mining balanced patterns from non-mirrors interactions). Let $\mathcal{T}$ be a transformed database obtained from a non-mirror interaction database $\mathcal{R}$. $\mathcal{T}$ is defined over $\mathcal{I}_1 \cup \mathcal{I}_2$ where $\mathcal{I}_k$ represents the type k of player ($k \in \{1, 2\}$) and $class : \mathcal{T} \rightarrow \{\mathcal{I}_1, \mathcal{I}_2\}$ assigns to any sequence its winner type. σ is a minimum frequency threshold. The problem is to extract the set of so-called *frequent balanced patterns* $\mathcal{F}_t$ such as for any $s \in \mathcal{F}_t$, $freq(s,\mathcal{T}^{\mathcal{I}_1}) \geq \sigma$ and $freq(s,\mathcal{T}^{\mathcal{I}_2}) \geq \sigma$ (implying $freq(s,\mathcal{T}) \geq \sigma$) and the balance measure is computed and given by:

$$balance(s, \mathcal{T}^k) = \frac{|sup(s, \mathcal{T}^k)|}{|sup(s, \mathcal{T}^1)| + |sup(s, \mathcal{T}^2)|}$$

Remark. The balance measure is a normalized version of the growth rate given in previous section such that $balance(.) \in]0, 1]$ and $balance(s, \mathcal{T}^1) + balance(s, \mathcal{T}^2) = 1$ which entails a zero-sum game property.

Example. Table 6.3 gives a transformed interaction database $\mathcal{T}$, obtained from a non-mirror interaction database $\mathcal{R}$, with $\mathcal{I}_1 = \{a, b\}$ and $\mathcal{I}_2 = \{c\}$ being the sets of actions of each player type. With $\sigma = 0.2$, $s = \langle\{ab\}\{c\}\rangle$ is a frequent balanced pattern since $freq(s, \mathcal{T}^{\mathcal{I}_1}) = \frac{2}{3}$ and $freq(s, \mathcal{T}^{\mathcal{I}_2}) = \frac{1}{2}$. Moreover, $balance(s, \mathcal{T}^{\mathcal{I}_1}) = \frac{2}{3}$ and $balance(s, \mathcal{T}^{\mathcal{I}_2}) = \frac{1}{3}$. It means that s wins two times more for the type 1 of player than for the type 2.

id	$s \in \mathcal{S}$
s_1	$\langle a^+\{b^+c^-\}\rangle$
s_2	$\langle a^+\{b^+c^-\}c^+\rangle$
s_3	$\langle d^+\{b^+c^+\}d^-\rangle$
s_4	$\langle a^-\{b^+c^+\}\rangle$

Table 6.4: A signed interaction database.

6.5 Balanced Patterns in Mirror Databases

In this section, we consider interaction sequence databases where the players have access to the same set of actions. Consequently, the latter cannot be partitioned in two sets and the previous approach can not apply. We propose a new interaction database representation, *signed interaction databases*. It enables to define *frequent balanced patterns* from an arbitrary interaction database (either mirror or non-mirror).

Definition (Signed interaction database). Recall that *Actions* is a finite set of *actions* shared by both players. We introduce $\mathcal{I}_s = Actions \times \{+, -\}$ denoting actions associated either to a positive or negative class. A signed database $\mathcal{S}$ is built from an interaction database $\mathcal{R}$ as follows: Each action of an interaction sequence is signed $+$ if it is performed by the winner and signed $-$ if performed by the loser (both players and class labels are dropped).

Definition (Dual of an item, itemset and sequence). Let $\mathcal{I}_s = Actions \times \{+, -\}$ be the set of signed items, or actions. For any $(a, c) \in \mathcal{I}_s$, also written a^c, we define its *dual* as

$$dual(a, c) = dual(a^c) = (a, \{+, -\} \backslash c) = a^{\{+,-\}\backslash c}$$

Informally, it means that the dual of a signed action is the same action where the class c has changed. This definition is simply propagated for itemsets and sequences of itemsets, for any $X \subseteq \mathcal{I}_s$ and any $s = \langle X_1, X_2, ...\rangle$ a sequence over $\mathcal{I}_s$:

$$\begin{aligned} dual(X) &= \{dual(x), \forall x \in X\} \\ dual(s) &= \langle dual(X_1), dual(X_2), ...\rangle \end{aligned}$$

Example. In Table 6.4, we have $\mathcal{I}_s = \{a, b, c, d\} \times \{+, -\}$, $dual(a^+) = a^-$ and $dual(s_1) = \langle a^-\{b^-c^+\}\rangle$.

These definitions enable now to naturally introduce a balance measure that would, for a sequential pattern s give the proportion of its support among the support of both itself and its dual.

Definition (Balance measure). Let s be a frequent sequential pattern in a database $\mathcal{S}$. The balance measure of s is

$$balance(s) = \frac{|sup(s, \mathcal{S})|}{|sup(s, \mathcal{S})| + |sup(dual(s), \mathcal{S})|} \tag{6.1}$$

This intuitive definition however does not hold. Since actions are shared by the two players, both a sub-sequence and its dual may occur in a single sequence $s \in \mathcal{S}$. Consider the following example: $\mathcal{S} = \{\langle\{a^+b^-\}\{a^-b^+\}\rangle, \langle\{a^-b^+\}\rangle\}$ with $\sigma = \frac{1}{2}$. The sequence $s = \langle\{a^+b^-\}\rangle$ is a frequent sequential pattern, and $|sup(s, \mathcal{S})| = 1$. We have also $|sup(dual(s), \mathcal{S})| = |sup(dual(\langle\{a^-b^+\}\rangle), \mathcal{S})| = 2$. Hence, $balance(s) = \frac{1}{1+2} = \frac{1}{3}$. However, since s and $dual(s)$ both appear in the first sequence, it should not be counted two times. This leads us to the definition of the balance measure in the general case in which we ignore sequences where both a pattern and its dual appear.

Definition (Generalized balance measure). For a sequential pattern s, the generalized balance measure is given by

$$balance_{gen}(s) = \frac{|sup(s,\mathcal{S}) \backslash sup(dual(s),\mathcal{S})|}{|sup(s,\mathcal{S}) \sqcup sup(dual(s),\mathcal{S})|} \tag{6.2}$$

where $\sqcup$ denotes the exclusive union $A \sqcup B = (A \cup B) \backslash (A \cap B)$. In the following, *balance* will always refer to the general version. We have that $balance(s) \in [0,1]$ and $balance(s) + balance(dual(s)) = 1$ which expresses a zero-sum game property.

Problem (Mining balanced patterns from signed interactions). Let $\mathcal{S}$ be a signed interaction database defined over $\mathcal{I}_s$ generated from an interaction database $\mathcal{R}$, and σ a minimum frequency threshold. The problem is to extract the set of so-called *frequent balanced patterns* $\mathcal{F}_s$ such as for $s \in \mathcal{F}_s$, $freq(s,\mathcal{S}) \geq \sigma$, $freq(dual(s),\mathcal{S}) \geq \sigma$ and the balance measure is computed and given by (2). Furthermore, the fact that both s and $dual(s)$ have to be frequent leads to redundant information: it is enough to keep s along with its support, balance measure and intersection of support $common(s) = |sup(s,\mathcal{S}) \cap sup(dual(s,\mathcal{S}))|$ to know the measures of its dual. As such, the problem is also to compute a non redundant collection of patterns $\mathcal{F}_s$ where, if $s \in \mathcal{F}_s$ then $dual(s) \notin \mathcal{F}_s$.

Example. Table 6.4 gives a signed interaction database $\mathcal{S}$ obtained from an arbitrary $\mathcal{R}$. With $\sigma = \frac{1}{4}$, $s = \langle a^+c^- \rangle$ appears two times, its dual appears only once, hence $balance(s) = \frac{2}{3}$.

Remark. Any interaction database, mirror or non-mirror, can be represented as a signed interaction database. For the non-mirror case, one can easily prove that for any balanced pattern s, we have $common(s) = \emptyset$ and thus Formula (1) holds.

6.6 Algorithms

We presented several algorithms whose description can be found in the original article [32]. All are based on the well-known framework for mining frequent sequential patterns, called Pattern-growth and its associated algorithm PrefixSPAN [133]. The most efficient algorithm we introduced, called BalanceSpan, introduces the notion of "double database projection" during the exploration: the first on the positive class, the other on the negative class. This allows a very efficient computation of the balance measure. We give in this manuscript the pseudo-code in Algorithm8 that we illustrate now.

Example. BalanceSpan operates on a signed interaction database. It proceeds to a double projection at a time. Let us still consider the dataset of Table 6.4 with $\sigma = 1/4$. Starting with the empty sequence and the entire dataset, the first step consists of finding frequent items: e.g., a^+ is frequent. BalanceSpan directly generates the dual of $\langle a^+ \rangle$ to proceed to the double projection. Thus, it checks if a^- is frequent and then projects on both $\langle a^+ \rangle$ and $\langle a^- \rangle$ at a time: i.e., it creates a new node in the pattern tree that is related to the couple $(\langle a^+ \rangle, \langle a^- \rangle)$. The next step calls *BalanceSpan* on this new node. So it searches for frequent items in the projected database $\mathcal{S}_{|\langle a^+ \rangle}$: e.g., c^-. Thus it checks if $dual(c^-)$ is frequent in $\mathcal{S}_{|\langle a^- \rangle}$: the node containing the couple $(\langle a^+c^- \rangle, \langle a^-c^+ \rangle)$ is created and explored in the next step.

6.7 Experiments

We study one of the most competitive real-time strategy games (RTS), StarCraft II (Blizzard Entertainment, 2010), successor of *StarCraft: Brood War*, test bed for many research in AI [128]. A game involves two players each choosing a faction among Zerg (Z), Protoss (P) and Terran (T): there are 6 different possible match-ups with different strategies of game. During a game, two players are battling on a map (aerial view), controlling buildings and units to gather supply, build an army with the final goal of winning by destroying the opponent's forces. Such actions (training, building, moving, attacking) are

Algorithm 8 BalanceSpan

Require: A sequence s, its dual sequence $s' = dual(s)$, the s-projected signed DB $\mathcal{S}_{|s}$, the s'-projected signed DB $\mathcal{S}_{|s'}$, and a minimum threshold σ
Ensure: The frequent balanced pattern set F
Compute $balance(s)$
$F \longleftarrow \{(s, s')\}$;
scan $\mathcal{S}_{|s}$ once, find every frequent item a such that
    (a) s can be extended to $(s \circ_i a)$, or (b) s can be extended to $(s \circ_s a)$
scan $\mathcal{S}_{|s'}$ once, find every frequent item b such that
    (a) s' can be extended to $(s' \circ_i b)$, or s' can be extended to $(s' \circ_s b)$
if no valid a or b available **then**
    return F;
end if
for valid a and b such that $a = dual(b)$ **do**
    (a) $F \longleftarrow F \cup BalanceSpan(s \circ_i a, s' \circ_i b, \mathcal{S}_{|s \circ_i a}, \mathcal{S}_{|s' \circ_i b}, \sigma)$
    (b) $F \longleftarrow F \cup BalanceSpan(s \circ_s a, s' \circ_s b, \mathcal{S}_{|s \circ_s a}, \mathcal{S}_{|s' \circ_s b}, \sigma)$
end for
return F;

done in real-time. Each faction (Z, P, T) allows different units and buildings with distinctive weaknesses and strengths following a rock-paper-scissors principle. As such, there are mirror match-ups (TvsT, PvsP, ZvsZ) and non-mirror match-ups (TvsP, TvsZ, PvsZ). A strategy is hidden in large sequences of actions generated by players and called *replays*.

Played as an electronic sport, StarCraft II is regularly patched: basic rules of the games are adjusted (properties of units, building times, ...), new rules are introduced through expansion sets (*heart of the swarm* and *legacy of the void*). The balance design team of StarCraft II, often needs to study historical data, care about player feedback on Web forums, and finally to justify their choices. After quantitative experiments of our algorithms, we will discuss the usefulness of our approach to help studying balance issues in RTS games.

6.7.1 Datasets

StarCraft II replays are files that store any action performed by all players during a game, and are easily accessible on the Web. We retained $91,503$ games with a total of 3.19 years of game time. The average length of a game is about 20 minutes. A game is selected if it involves a high level players (in the highest leagues and playing at least 200 actions per minute), since casual (by opposition to professional) players are not able to follow specific strategies. We divided the $91,503$ replays into six different sequence datasets, one for every match up. Buildings are one of the key elements of a strategy, since they enable different kinds of units production: from each replay, we derive a sequence where the items represent the buildings the players chose to produce in real time, and itemsets denote time windows of 30 seconds. We consider only the 10 first minutes of each game (the strategical impact of a building is less important after 10 minutes). Table 6.5 summarizes all characteristics of the datasets.

6.7.2 Exploration of the extracted patterns

It is interesting to visualize the distribution of both the support and the balance of the patterns. Figure 6.1 gives such distribution for dataset ZvZ that enables very fast computations with low σ (less than 5 seconds for $\sigma = 0.001$). There, both a pattern and its dual are presented, which allows interestingly to observe that the equation $y = 0.5$ (where y is the vertical axis) gives almost a symmetry axis. Indeed, both a pattern and its dual do not necessarily have the same support. One can notice that empirically, there are high chances for a pattern with high frequency to have a fair balance around 0.5. This behavior applies

Dataset	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	s_{max}	s_{avg}	i_{max}	i_{avg}
$\mathcal{S}$ - PvP	6,823	26	42	21.7	6	1.8
$\mathcal{S}$ - PvT	19,270	62	42	25.9	8	1.8
$\mathcal{S}$ - PvZ	23,491	52	41	23.1	7	1.7
$\mathcal{S}$ - TvT	7,598	36	38	23.8	7	1.8
$\mathcal{S}$ - TvZ	24,459	62	37	21.2	8	1.6
$\mathcal{S}$ - ZvZ	9,922	26	28	10.4	7	1.4
$\mathcal{T}$ - PvT	19,270	34	43	26.9	8	1.8
$\mathcal{T}$ - PvZ	23,491	29	42	24.1	7	1.7
$\mathcal{T}$ - TvZ	24,459	34	38	22.2	8	1.6

Table 6.5: Datasets: sequence and item counts; max. and avg. sequence sizes; max. and avg. itemsets sizes.

for the other dataset and is what we could expect given the definition of the balance measure.

It is possible to query the set of extracted patterns in various ways. Indeed, the pattern mining task is related to the Knowledge Discovery in Databases (KDD) process that aims at extracting knowledge from data [59]. Data mining approaches and more precisely pattern mining approaches are a step of the KDD process that results in patterns from transformed data [5]. Our work is a pattern mining approach to study strategy balance that is applied to RTS games. Thus, one gets a set of patterns that are a generalization of the local strategies within the data. BalanceSpan still requires an analysis once the patterns are outputted.

Exploring a large collection of patterns can be done in many ways. First, as illustrated hereafter, the expert can filter the collection with specific constraints such as a minimum number of itemsets, or specific items using regular expressions, etc. Second, the expert can introduce preferences as measures on the patterns (size, length, support, balance, etc.) that he wishes to minimize or maximize given his goals (the so-called *skypatterns* [148]). Indeed, one expert may favor highly balanced patterns with high support (probably the standard strategies), while another could be interested in maximizing the support while favoring patterns whose balance is closest to 0.5 (giving hints to possible game design problems). Finally, the discovery can be done interactively, through an interactive algorithm (not only the full KDD process [158]). The basic assumption is that the expert does not really know what he is looking for in the data, and guides the pattern discovery at each step of the algorithm (such as sequence expansion in our case). In all cases, the pattern language must be clearly defined and efficient algorithms proposed to compute the measure of interest, our main contribution.

We now provide a basic example of exploration by querying the pattern set. Out of the $43,610$ patterns for ZvZ with $\sigma = 0.001$, we can keep the patterns involving two players (containing both $+$ and $-$), which returns $40,674$ patterns. Then we restricted the set of patterns to those involving two specific items (RoachWarren and Spire) to get only 290 patterns. $\langle SpawPool^{+}, SpawnPool^{-}, SpiCrawler^{+}, RoachWarren^{+}, Spire^{-}\rangle$ denotes for example games where one of the players go to air units and the second to ground units, two known openings, with balance 0.47 and support 68.

The pattern mining task can be adapted in various ways, depending on which and how the basic actions are encoded in the sequence. Let us now sketch different scenarios.

6.7.2.1 Discovering strategy openings

Openings are the most well-known strategies and executed during the first five to ten minutes of a StarCraft II game. It is expected that openings are balanced to make the game enjoyable for the casual player, competitive for the professional, but also interesting to watch for the spectators [45]. We build our sequence databases with a set of items composed of tuples $(building, sign, i^{th}window)$, with fixed windows of 30 seconds by default, i.e., the i^{th} window contains the items performed between the $((i-1)\times 30)^{th}$ second and the $((i\times 30)-1)^{th}$ second. We expect that openings are found as the more frequent patterns and being also balanced: Figure 6.2 shows the complete set of patterns for the ZvZ dataset which differs with Figure 6.1 by its skewness. We explored the resulting patterns with a game expert. When considering another dataset (PvZ), we obtain only 591 patterns with $\sigma = 0.05$. Top frequent patterns represent all

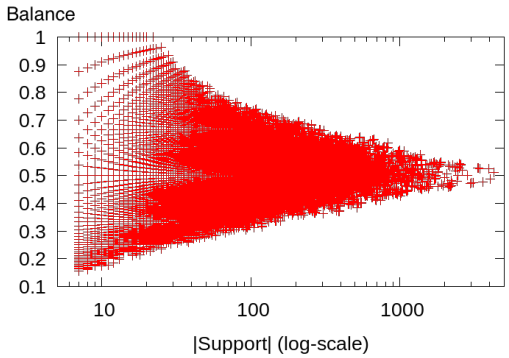


Figure 6.1: Patterns support and balance for the dataset ZvZ.

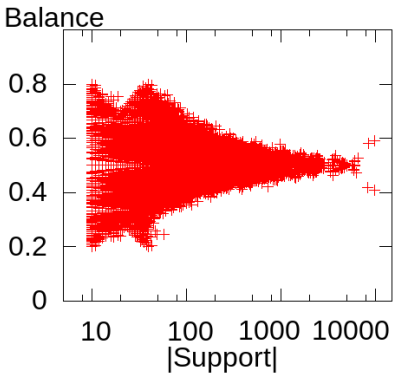


Figure 6.2: Game openings support and balance for the dataset ZvZ.

well-known strategies: $s = \langle\{(Nexus, +, 5\ )\}\ \{(Gateway, +, 6\ )\ (Photon\ Cannon, +\ 6\ )\}\rangle$ represents a popular Protoss strategy, no matter the strategy of the opponent is. It is balanced ($balance(s) = 0.52$).

6.7.2.2 Discovering possible balance issues (hypotheses elicitation)

The rules of the game are set by the editors and developers. However, such rules are not always fair and balanced, and such weaknesses can only be discovered after weeks. We asked an expert to highlight a well known imbalanced strategy. The so called *bunker rush* was used a lot by Terran players against their Zerg opponents. It consists of building in the early stage of the game a *bunker* near the opponent's base to put his economy and development in difficulty. After several complaints from the StarCraft II community, the rules changed on 10^{th} May 2012: a Zerg counter unit (*the queen*) has been slightly improved. Since then, this strategy stopped to be used for some time. Our approach should be able to reflect/discover that fact: we proceed as follows. We split dataset TvZ into two parts: the first one, called $\mathcal{S}_{before}$, contains replays that happened strictly before 10^{th} May 2012 (17,171 replays), and the second one, called $\mathcal{S}_{after}$, contains the replays that happened strictly after this change (6,698 replays). The mining of the dataset $\mathcal{S}_{before}$ (respectively $\mathcal{S}_{after}$) with a low $\sigma = 0.05$ returns 8,138 patterns (resp. 7,735). According to the experts, the *bunker* should be built during the sixth window of time (between 2'30" and 3' of the game). There are 20 (resp. 12) patterns that involve the item $(Bunker, c, 6)$ with $c \in \{+, -\}$. With $\mathcal{S}_{before}$ (resp. $\mathcal{S}_{after}$), the average value of the balance is 0.58 (resp. 0.51) with a standard deviation equals to 0.5 (resp. 1.6). This is clear that since the patch was released, this strategy has become balanced. Moreover, we can remark that this strategy is no longer used by players: in fact the number of extracted patterns related to this strategy decreases by 40% whereas the number of extracted patterns only decreases by 5% from $\mathcal{S}_{before}$ to $\mathcal{S}_{after}$. Thus, BalanceSpan enables to see the impact of the release of patch, by analyzing the period before and after this key date.

6.7.2.3 On the diversity in mirror match-ups

It delicate to speak about balance when both players belong to the same faction (mirror match-ups): both players have access to the same strategies (same building, hence a symmetrical game). Let us observe for example the dataset PvP with a new vocabulary: items are tuples $(building, sign, i^{th}window,\ j^{building})$ where the last element records how many of this building were already made at the moment of the action (cumulative). With a minimal support $\sigma = 0.05$, we obtain 3418 patterns. We can find here the so-called *4 Gates* strategy through the pattern $s = \langle\{(Gateway, +, 3, 1)\ (Assimilator, +, 3, 1)\}\ \{(Cyb.Core, +, 4, 1)\}\ \{(Gateway, +, 7, 2)\ (Gateway, +, 7, 3)\ (Gateway, +, 7, 4)\}\rangle$ with $balance(s) = 0.59$. Such a high value may be surprising, but it reflects the effectiveness of this strategy, and consequently the poor diversity of the

strategies used in the PvP match-up. It is an easy and non-risky strategy to apply: according to the expert, a player has better chances to win with this strategy against riskier strategies. After recurrent complaints, a major update of the game introduced new units and buildings: Since then strategies used in PvP are more diversified and the *4 gates* strategy is rarely used.

6.8 Related Work

Discovering patterns that highly distinguish a dataset from others (e.g., *win* labeled objects versus *lose* labeled objects) is an important task in machine learning and data mining [127]. One of the main reason is that such patterns enable the building of comprehensible classifiers [65]. In the general settings, we are given a set of objects of different classes that take descriptions, generally from a partially ordered set (itemsets, graphs, intervals, etc.) [98]. The goal is to find good description generalizations that mostly appear in one class of objects and not in the others. In different fields of AI and applied mathematics, such descriptions have different names [127] like *version spaces* [121], *contrast sets* [21] and *subgroups discovery* [79] in machine learning, *emerging patterns* [52] in data-mining; or *no counter-example hypothesis* in formal concept analysis [98]. Our contribution in this field is to consider and compute efficiently a balance measure that existing methods can partially or non efficiently compute.

StarCraft II and other real time strategy games (RTS) in general, face several research challenges in artificial intelligence as deeply discussed in a recent survey [128]. Our work is related to the challenge that the authors of the survey qualify as *prior learning*, that is, techniques that can *"exploit available data such as existing replays [...] for learning appropriate strategies beforehand"*. Strategies in RTS game are complex and divided in several tasks, each bringing difficult problems. Several case-based reasoning approaches have been proposed, mainly to retrieve and adapt strategies (especially build orders) to be used then by an automated agent [9, 163]. Other kinds of approaches are also used for several prediction tasks. Predicting the opponent's production was studied with answer set programming [149], while learning transition probabilities within build orders was achieved with hidden Markov models [50]. Weber et al. described any past game by a vector of building and upgrade timings: such features allow an accurate strategy prediction [164]. This comes with a limit: game logs are *a priori* labeled with a strategy using rules based on "manual" expert analysis. The same applies for opening prediction [151].

To discover strategies in large volumes of replays, avoiding to manually label game logs, knowledge discovery in database (KDD) methods are required, and especially pattern mining techniques. This was highlighted in the open problems category *Domain knowledge* of the recent survey mentioned before [128]: *"Is it possible to devise techniques that can automatically mine existing collections [...] and incorporate it into the bot?"*. We did not study the second step (incorporation), but presented a way to extract efficiently such patterns and focused on *balance* issues for helping game designers. A long road remains to be able to select the right patterns to be used by artificial agents as discussed recently in [103, 128].

6.9 Conclusion

We tackled the problem of mining frequent sequential patterns in real time strategy games whose balance measures provide meaningful insights on the strategies played and their ability of being in equilibrium or not. For that matter, we revisited the well known notions of discriminant pattern mining to provide efficient algorithms for the elicitation of balance hypotheses from the data. From that, we presented several algorithms that enable dealing with interaction databases, and we showed that only BalanceSpan enables to deal with all datasets efficiently. We empirically validated that the balance measure is able to distinguish balanced and imbalanced strategies. We believe that our approach can become a basic tool for *balance designers* when analyzing a subset of historical data of a game in beta phase, or even after its release, through an exploratory process (KDD and interactive mining). A major difficulty remains to select and construct features of interest from the game logs; it is part of the general KDD task.

Chapter 7

Revealing the Identity of Anonymous Players

Video game is a very lucrative industry, unleashed by the ubiquity of gaming devices, multi-player networks and live broadcasting platforms. Games generate large amounts of behavioural data which are valuable to face the new challenges of *video game analytics* such as detecting balance issues, bugs and cheaters. In electronic sports (e-sports), cyberathletes conceal their online training using different aliases or avatars (virtual identities), which allow them not being recognized by the opponents they may face in future competitions (with cash prices challenging already most of the traditional sports). It was recently suggested that behavioural data generated by the games allows predicting the avatar associated to a game play with high accuracy. However, when a player uses several avatars, accuracy drastically drops as prediction models cannot easily differentiate the player's different avatar aliases. Since mappings between players and avatars do not exist, we introduce the *avatar aliases identification problem* and propose an original approach for alias resolution based on supervised classification and Formal Concept Analysis. We thoroughly evaluate our method with the video game *Starcraft 2* which has a very wide and active community with players from diverse cultures and nations. We show that under some circumstances, the avatars of a given player can easily be recognized as such. These results are valuable for e-sport structures (to help preparing tournaments), and game editors (detecting cheaters or usurpers).

7.1 Introduction

Currently, video games are a popular and lucrative industry generating more revenue than both Cinema and Music. This recent and fast development has been catalysed by several factors. First, an increasing part of the population has access to connected electronic devices (computers, smart-phones, etc.), and to the Web including a myriad of single and/or multi-player games. Video games are now a popular leisure activity. Furthermore, the recent development of competitive gaming and live video game streaming platforms [93] makes electronic sports (e-sports) a reality [152]. Professional players on contract with teams and sponsors are taking part in competitions with cash prices challenging already most of the traditional sports. Cyberathletes are widely followed in so-called *off-line events* and daily supported on the Web through their own live broadcasts [93] generating the fourth largest stream of data in 2015 for the platform, *Twitch.tv*. People not only enjoy playing, but also enjoy watching the others for a variety reasons [45, 93]. Games generate tremendous amounts of behavioural data, valuable to face the many new industrial challenges such as detecting balance issues [32], bugs and cheaters [161], but also valuable for several academic fields such as artificial intelligence [128], computer human interactions [170] and cognitive sciences [154].

In this paper, we introduce a new problem: the *avatar aliases identification problem*. In most of online competitive games, players need an "avatar" (i.e., an online identity) to log in the game network.

Nothing forbids a player to have several avatars and actually, it is a very common practice for professional players, or cyberathletes, of several video game franchises (e.g. *League of Legends*, *Starcraft 2*, *Dota 2*, *Counter Strike*). Players generally have one official avatar for official tournaments, and several others to conceal their tactics while playing without being recognized by other players they may meet online: global rankings and leagues are public just as in chess and tennis, while game logs (called replays) are available and prone to analysis by means of visualization[12], machine learning and data mining techniques [109, 151] just as in standard sport analytics. Accordingly, we are facing a set of players, generating behavioural data (game logs or traces), in a one-to-many relationship with avatars (the many-to-many relationship will be discussed later). Our goal is not to discover this mapping, since players privacy is protected. Instead, the *avatar aliases identification problem* aims at discovering groups of avatars belonging to the same player. Solving this problem is motivated by the growing need of e-sport structures to study the games and strategies of the opponents (tournament preparation), and the security challenges of game editors (detecting usurpers).

Recently, it was suggested that behavioural data hide individual identifying patterns [154, 170], (tested on the real time strategy game *Starcraft 2* (©Blizzard)). After choosing features related to keyboard usage, Yan et al. showed that a classifier can be trained to predict with high accuracy the avatars involved in a game play [170]. Nevertheless, they purposely considered datasets without players having several avatars (what we call avatar aliases). Indeed, in presence of such *avatar aliases*, the prediction accuracy drastically degrades, since prediction models fail at differentiating two avatars of the same player. We build on this work by proposing an approach called DÉBROUILLE for answering the *avatar aliases identification problem*: it relies on mining confusion matrices yielded by a supervised classifier using Formal Concept Analysis [64], and exploits the confusion a classifier has in presence of avatar aliases when they belong to the same player. Experimental evaluation shows very good results under certain restrictions imposed to the set of players involved in a dataset.

The remainder of this chapter is organized as follows. Section 7.2 introduces the problem settings, our claims and goals. We formally present an original methodology in Section 7.3 for discovering avatar aliases and the evaluation strategy in Section 7.4. Our results are assessed through an extensive evaluation in Section 7.5. Related work is discussed in Section 7.6 before we conclude.

7.2 Problem Settings

7.2.1 *Starcraft 2* in competitive gaming

We study the *real-time strategy* game (RTS) *Starcraft 2*, a franchise released in 2010 which met success in competitive gaming and e-sports [152]. A standard game involves two players and each chooses a faction (Zerg, Protoss or Terran) with different weaknesses and strengths (following a rock/paper/scissors principle). Players control buildings and units through an aerial view of the battleground map, collect resources to build an army and achieve victory by destroying the opponent's forces. A *player* needs an *avatar* to enter the dedicated multi-player system called `Battle.net`. In the context of e-sport, our careful attention is given to the *avatar aliases identification problem* that affects the game[13]. Nothing forbids a user from having multiple avatars, which is actually a common practice among professional players for a variety of reasons. For example, as they compete in international tournaments, they may need an avatar in each different *Battle.net* server (USA, Europe, Korea, etc.). Another important reason is that professional users may need to hide their tactics while training before competitions. For this reason, cyberathletes may create different avatars with names such as `lIlIlIlIlIlll` called *bar code avatars* (as in Table 7.3) for practising on public servers without being recognized. Practise on public networks corresponds to a major time of their activity [152]. Currently, among the best 50 avatars in the global *Starcraft 2* ranking system, 46 avatars are *bar code*[14]. As argued in the introduction, multi-aliases prompt several problems. Our goal is to formalize and solve the *avatar aliases identification*

[12] For example, `https://sites.google.com/site/sc2gears/` for *Starcraft 2*
[13] Multi-aliases are known as "smurfs" players within the gamer community.
[14] `http://www.sc2ranks.com` visited on 2015, May 28^{th}

problem which consists in finding avatars belonging to the same player without any information about the individual but his game behaviour.

7.2.2 Behavioural data

During a game, a *player* performs a series of *actions* which generates a *trace*. Basic actions rely on mouse usage, provided with semantics: "selection of a unit", "attack with selected unit", "collect resources with selected unit", etc. Many (if not all) of these actions can be performed with the mouse by selecting a series of menus on the screen. However, to improve the amount of actions a user can perform per minute (a critical issue in this kind of video games), expert players prefer to bind these actions directly to the keyboard through the use of customized *hotkeys*, also called *control groups* [170]. As such, any action, done through mouse or keyboard, is stored in *replay files* that are available on the Web and usable for observing and studying other players strategies. Replays constitute extremely rich behavioural data, and we use them for answering our problem. Table 7.1 shows two traces associated to a single game: Player *TAiLS* starts by selecting with the mouse a building called *Base*, assigns it to hotkey 1 (a), selects some units with the mouse (s), select units attached to hotkey 2 (s), etc.

Avatar	Game trace	Outcome
`RorO`	s,s,hotkey4a,s,hotkey3a,s,hotkey3s, ...	Lose
`TAiLS`	Base,hotkey1a,s,hotkey1s,s,hotkey1s, ...	Win

Table 7.1: Traces of a match for two players

7.2.3 Predicting avatars from behavioural traces

Let us consider the following scenario. Given a trace, is it possible to find its associated avatar? In Table 7.1, given the trace in the first row, is it possible to say that the avatar associated to it is `RorO`? We can approach this question from a classification point of view. Given a training set of traces labelled with their associated avatars, we would like to classify new "instances" (new traces) in the space of classes represented by the set of avatars. As such, we can apply any supervised classification technique in this direction. Recently, Yan et al. [170] suggested that behavioural data, as presented in the previous subsection, offer predictions with high accuracy, when there are no aliases in the dataset. In presence of aliases however, the accuracy drastically drops, as trained models are not able to differentiate properly different avatars of a same individual.

The reason why game traces allow good avatar predictions, when there are no aliases, is related to the cognitive process involved in the activity of playing [154]. A game of *Starcraft 2* is short (between 15 and 20 minutes in average), in real time, and several tasks and strategies are repeated thousands of time to be achieved instinctively: mastering the game requires an intense multi-tasking activity (not sequential) and daily practise. Novice's traces will seem messy during the learning process, but will converge with practise towards distinctive patterns, or individual *signatures*.

7.2.4 Towards confusion clustering to identify avatars aliases

In general, players use several avatars that generate traces. This is modelled in Figure 7.1. The mapping $owns : Player \rightarrow Avatar$ is however not known (or very partially by the game editor that keeps it private). The only information available is the mapping $generates : Avatar \rightarrow Trace$. Notice that we make the assumption that an avatar belongs to a single player (individual). This may not hold: nothing forbids two persons from sharing an avatar. However, as an avatar is ranked according to *all* its games (wins and loses) in a world ranking system, it is fairly safe to assume in general that professional players (our focus) do not share their avatars to protect their *reputations* but also *privacy* when playing in secrecy.

The general intuition of our approach to tackle the *avatar aliases identification problem* is now introduced. Our model can be split in two sub-tasks, namely "finding Trace patterns associated to Avatars"

and "finding Avatar clusters associated to Players". For the first task, a classifier is built as explained in the previous subsection, for predicting the avatar involved in a game. In presence of avatar aliases, its accuracy drastically drops. However, it is a fair hypothesis that the classifier confusion shall be spread locally among avatar aliases. As such, the second step involves a particular clustering of the confusion matrix, that outputs avatar aliases candidates. We formalize this step with FCA: a fuzzy set operator on class memberships enables the finding of candidate patterns.

Figure 7.1: A simple model of game traces generations

7.3 Mining a Confusion Matrix with FCA

Let A be a set of avatars and T be a set of traces such that for a given avatar $a \in A$, the set $T_a \subseteq T$ is the set of all traces generated by a. Consider a classifier ρ where labels are the avatars to predict. Our method consists in using the confusion matrix generated by a classifier ρ and analyse how confusion between labels is spread to extract candidates of avatar aliases. This process has two steps: firstly, specific patterns are extracted from the confusion matrix, secondly, they are scored, ranked and post processed. The corresponding pseudo-code of our approach, called DÉBROUILLE for *unscramble* in French, is given in Algorithm 9 and is detailed hereafter.

7.3.1 Classifying Traces

A classifier is a function $\rho : T \to A$ that assigns the avatar $\rho(t) \in A$ to a given trace $t \in T$. Let $n = |A|$ be the number of avatars in A, from any classifier ρ, one can derive a confusion matrix

$$C^{\rho}_{n \times n} = (c_{i,j})$$

where

$$c_{i,j} = |\{t \in T_{a_i} \ s.t. \ \rho(t) = a_j\}|$$

Each row and column of C^{ρ} correspond to an avatar, while the value c_{ij} is the number of traces of avatar a_i that are classified by ρ as of avatar a_j. The normalized confusion matrix is given by

$$\tilde{C}^{\rho} = [c_{i,j}/|T_{a_i}|]$$

Algorithm 9 DÉBROUILLE pseudo code

Require: $\tilde{C}^{\rho}$: normalized confusion matrix, λ cluster score threshold, s score threshold.
Ensure: P list of pairs of avatar ranked by cluster score.

```
P ← ∅
C ← MineFuzzyConcepts(C̃^ρ, s)
C ← rank C according to s
for all c ∈ C do
    pairs ← pairs from the pattern concept extent of c
    for all p ∈ pairs do
        if p ∉ P and cluster_score(p) > λ then
            P ← P ∪ {p}
        end if
    end for
end for
P ← rank P according to the cluster score
return P
```

where $\tilde{C}^{\rho}_{i,i} = 1$ for any $i \in [1, |A|]$ means all the traces of avatar a_i are correctly classified by ρ.

Table 7.2 gives an example of normalized confusion matrix between five avatars. It is worth mentioning that we have generalized the classifier to a given function ρ and thus, we will treat it as a "black box" with an input (in the form of a set of traces T and avatars A) and an output which corresponds to the confusion matrix. Internally, features built from traces can be chosen independently, and the classifier can split the set of traces into train and test sets using different strategies.

7.3.2 Clustering Avatars

Our goal is to discover groups of avatars that belong to the same player. Our intuition is that a classifier will hardly differentiate these avatar aliases, hence the confusion matrix values should be high and concentrated around them. This is exemplified in Table 7.2: avatars $\{a_1, a_2\}$ are candidates to belong to the same player, $\{a_4, a_5\}$ shall belong to another user, while a_3 stays as singleton with a diagonal high value. Hence, a reasonable clustering of avatars would be given by the partition $\{\{a_1, a_2\}, \{a_3\}, \{a_4, a_5\}\}$.

Table 7.2: Confusion Matrix example

	a_1	a_2	a_3	a_4	a_5
a_1	0.6	0.4	0	0	0
a_2	0.4	0.55	0.05	0	0
a_3	0	0	0.8	0.15	0.05
a_4	0	0.05	0	0.7	0.25
a_5	0	0	0	0.5	0.5

More formally, given a normalized confusion matrix $\tilde{C}^{\rho}$, we would like to find pairs of avatars $a_i, a_j \in U$ such that $\tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{ji} \simeq \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{jj}$ and $\tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ji} + \tilde{C}^{\rho}_{ii} + \tilde{C}^{\rho}_{jj} \simeq 2$. These conditions come from the fact that, if a_i, a_j correspond to the same player, traces in T_{a_i} have the same probability of being classified as a_i or a_j (the same for traces in T_{a_j}). Furthermore, for a trace of avatar a_i, it is required that the probability of classification is spread between a_i and a_j only, meaning that $\tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ii} \simeq 1$ (similarly for a_j).

Using this rationale, in what follows we propose (i) to extract patterns from the confusion matrix, and (ii) to post process them to provide groups of candidate avatar pairs. The first step is achieved thanks to Formal Concept analysis (FCA [64, 62]), while we define scoring functions and ranking for the second step.

7.3.2.1 Fuzzy pattern structure

Let us define the fuzzy set of membership degrees L^A where $L = [0, 1]$, such as the mapping function $\delta : A \to L^A$ assigns membership values for the avatar a_i in the fuzzy set L^A based on the normalized confusion matrix. Simply, this is a mapping that assigns to a_i its corresponding row in $\tilde{C}^{\rho}$ which we denote $\tilde{C}^{\rho}_i$.

We model accordingly a confusion matrix $\tilde{C}^{\rho}$ as a pattern structure $(A, (L^A, \sqcap), \delta)$. The operator $\sqcap$ is a meet operator in a semi-lattice (idempotent, commutative and associative), and is defined as follows, given two avatars $a_i, a_j \in A$:

$$\delta(a_i) \sqcap \delta(a_j) = \langle min(\tilde{C}^{\rho}_{ik}, \tilde{C}^{\rho}_{jk}) \rangle,\ k \in [1, |A|]$$
$$\delta(a_i) \sqsubseteq \delta(a_j) \iff \delta(a_i) \sqcap \delta(a_j) = \delta(a_i)$$

Actually, $\sqcap$ corresponds to the fuzzy set intersection and $(L^A, \sqsubseteq)$ is a partial order over the elements of L^A which can be represented as a semi-lattice.

The pattern structure $(A, (L^A, \sqcap), \delta)$ is provided with two derivation operators, forming a Galois connection [62]. Formally, we have, for a subset of avatars $\mathtt{A} \subseteq A$ and a fuzzy set $\mathtt{d} \in L^A$ such as:

$$\mathtt{A}^{\square} = \prod_{a \in \mathtt{A}} \delta(a) \qquad \mathtt{d}^{\square} = \{a \in A \mid \mathtt{d} \sqsubseteq \delta(a)\}$$

A pair $(\mathtt{A}, \mathtt{d})$ is a pattern concept iff $\mathtt{A}^{\square} = \mathtt{d}$ and $\mathtt{d}^{\square} = \mathtt{A}$. Pattern concepts are ordered by extent inclusion such that for $(\mathtt{A}_1, \mathtt{d}_1)$ and $(\mathtt{A}_2, \mathtt{d}_2)$ we have:

$$(\mathtt{A}_1, \mathtt{d}_1) \leq (\mathtt{A}_2, \mathtt{d}_2) \iff \mathtt{A}_1 \subseteq \mathtt{A}_2 \,(\text{or } \mathtt{d}_1 \sqsupseteq \mathtt{d}_2)$$

Intuitively, a pattern concept $(\mathtt{A}, \mathtt{d})$ contains a fuzzy set $\mathtt{d}$ which can be represented as a *vector* $\mathtt{d} = \langle \mathtt{d}^j \rangle$ with length $|A|$ where each value $\mathtt{d}^j$ is the minimum for all rows i in column j of matrix $\tilde{C}^{\rho}$ such that $a_i \in \mathtt{A}$.

Example. The Table 7.2 illustrates a confusion matrix involving five avatars. It has been obtained from a classifier ρ. We illustrate first how pattern concepts are generated:

$$\begin{aligned} \delta(a_1) &= \{a_1^{0.6}, a_2^{0.4}, a_3^{0}, a_4^{0}, a_5^{0}\} \\ \delta(a_2) &= \{a_1^{0.4}, a_2^{0.55}, a_3^{0.05}, a_4^{0}, a_5^{0}\} \\ \delta(a_1) \sqcap \delta(a_2) &= \{a_1^{0.4}, a_2^{0.4}, a_3^{0}, a_4^{0}, a_5^{0}\} \end{aligned}$$

7.3.2.2 Scoring concepts and extracting aliases

The scoring function $s : L^A \to [0, 1]$ is given as follows: for a pattern $\mathtt{d}$,

$$s(\mathtt{d}) = \sum_{j=1}^{|A|} \mathtt{d}^j$$

It is clear that function s is decreasing w.r.t. the order of pattern concepts, i.e. $(\mathtt{A}_1, \mathtt{d}_1) \leq (\mathtt{A}_2, \mathtt{d}_2) \implies s(\mathtt{d}_1) \leq s(\mathtt{d}_2)$. Thus, pattern concepts can be mined up to a given score threshold analogously as formal concepts can be mined up to a given minimal support, as it is done in pattern mining [69]. We can appreciate that the higher the score of a given pattern, the more *confused* is the classification of traces of avatars $a \in \mathtt{A}$ by ρ in $\tilde{C}^{\rho}$ and thus, they become candidates for merging. This property directly follows from the choice of our similarity operator $\sqcap$ as a fuzzy set intersection, which behave as a pessimistic operator (returning minimum values).

The pattern mining step is executed as follows and corresponding to the *MineFuzzyConcepts* step (Line 2) in Algorithm 9. From the confusion matrix we compute all possible pattern concepts using the `addIntent` algorithm [157]. Pattern concepts are then ranked according to their score (Line 3) and converted into a list of pairs (Line 5). For example, if a pattern concept extent contains three avatars a_1, a_2 and a_3, we convert this concept into pairs (a_1, a_2), (a_1, a_3) and (a_2, a_3). The order among pairs of the same pattern is disregarded.

The `addIntent` algorithm is known to have a linear complexity w.r.t. the number of possible pattern concepts [157] which can grow exponentially w.r.t. the number of avatars in A. In our case, experimental evidences suggest that the number of pattern concepts in this setting is much lower than A^2, given the empty intersection between most pairs of avatars in A. Finally, given that the scoring function is monotonous w.r.t. the order $\sqsubseteq$, it can be used as a filter to stop the calculation of meaningless patterns.

Example. Continuing the previous example, we have:

$$s(\{a_1, a_2\}^{\square}) = 0.8 \tag{7.1}$$

$$s(\{a_4, a_5\}^{\square}) = 0.75 \tag{7.2}$$

$$s(\{a_1, a_2, a_4\}^{\square}) = 0.05 \tag{7.3}$$

7.3.2.3 Post-processing candidates

Consider the clustering condition previously formalized as $\tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{ji} \simeq \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{jj}$ and $\tilde{C}^{\rho}_{ii} + \tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ji} + \tilde{C}^{\rho}_{jj} \simeq 2$. Consider that the pair of avatars (a_i, a_j) respects these conditions. It is easy to see that (a_i, a_j) will necessarily be a candidate pair highly ranked from the previous step.

$$\begin{aligned} \tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{jj} \simeq min(\tilde{C}^{\rho}_{ij}, \tilde{C}^{\rho}_{jj}) \\ \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{ji} \simeq min(\tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ji}) \\ \implies min(\tilde{C}^{\rho}_{ij}, \tilde{C}^{\rho}_{jj}) + min(\tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ji}) \simeq 1 \end{aligned}$$

Thus, the set of avatar clusters we are looking for are contained within the set of candidate pairs and moreover, they are highly ranked. In order to filter the list of candidates from pairs that do not hold the avatar cluster definition, we propose a cosine similarity measure between a couple of vectors calculated for each avatar as follows. Let (a_i, a_j) be a candidate pair, the cluster score is defined as:

$$cluster\_score(a_i, a_j) = cosine(\langle \tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ij} \rangle, \langle \tilde{C}^{\rho}_{jj}, \tilde{C}^{\rho}_{ji} \rangle)$$

The cluster score establishes a measure of how close is a candidate pair from being an avatar cluster. The logic of this follows from the following scenario. Consider that the traces of avatar a_i were all correctly classified meaning that $\tilde{C}^{\rho}_{ii} = 1$ and that the traces of avatar a_j were all incorrectly classified as a_i, meaning that $\tilde{C}^{\rho}_{ji} = 1$, thus we have the following section of the normalized confusion matrix:

	a_i	a_j
a_i	1	0
a_j	1	0

We can observe that the pair (a_i, a_j) will be contained in the set of candidate pairs and will be highly ranked, even though it is not an avatar cluster since it violates the first condition. The cluster score for this particular case can be calculated as:

$$cluster\_score(a_i, a_j) = cosine(\langle 1, 0 \rangle, \langle 0, 1 \rangle) = 0$$

meaning that this candidate pair is not an avatar cluster. Notice that for the pair of avatars such that $a_{ii} = 1$ and $a_{jj} = 1$, the cluster score is 1 (cosine between parallel vectors) while the pair is not an avatar cluster. Indeed, this is true, however this pair would have a score s equal to 0 and would be at the bottom of the ranked candidate pairs. A third kind of pair occurs when the traces of a_i and a_j are all incorrectly classified as a third avatar a_k. In such a case, the cluster score is 0.

The post processing step is executed as follows as depicted in Algorithm 9. Given a ranked list of candidate pairs yielded from the previous step (Line 2 and 3), each pair is evaluated using the cluster score. Given an arbitrary threshold λ, if the cluster score of the candidate pair is below this threshold, then it is rejected (Line 7 and 8). Candidate pairs are re-ranked into a final list of avatar clusters (Line 9).

7.4 Evaluation

In this section, we provide a detailed evaluation procedure used in the experiments for assessing the ability of our approach at finding avatar aliases.

7.4.1 Avatars matching

As we do not have information about the users behind the avatars, it is not possible to actually evaluate the candidate pairs for merging using a "ground truth". Instead, we perform an indirect evaluation of our approach using three different strategies one being specific to the game *Starcraft 2*. Indeed, as illustrated in Figure 7.2, the avatar system of *Starcraft 2* is more elaborated than our general model given in Figure 7.1. The Table 7.3 exemplifies this model.

User	Descriptor
Account: (eu,2452136)	
Avatar	**URL**
MinChul	http://eu.battle.net/sc2/en/profile/2452136/1/MinChul/
SKMC	http://eu.battle.net/sc2/en/profile/2452136/1/SKMC/
Account: (eu,4233584)	
Avatar	**URL**
INnoVation	http://eu.battle.net/sc2/en/profile/4233584/1/INnoVation/
lIlIllIlIlll	http://eu.battle.net/sc2/en/profile/4233584/1/lIlIllIlIlll/
Account: (us,288081)	
Avatar	**URL**
Minigun	http://us.battle.net/sc2/en/profile/288081/1/Minigun/
ROOTMinigun	http://us.battle.net/sc2/en/profile/288081/1/ROOTMinigun/
Account: (us,2929052)	
Avatar	**URL**
ROOTheognis	http://us.battle.net/sc2/en/profile/2929052/1/ROOTheognis/
Account: (us,3023756)	
Avatar	**URL**
MinChul	http://us.battle.net/sc2/en/profile/3023756/1/MinChul/

Table 7.3: An example of five Battle.net accounts and their respective avatars

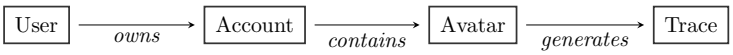


Figure 7.2: The trace generation model in *Starcraft 2*

Nicknames Each avatar is associated with a non-unique (nick-)name. Names are chosen by users and **can** be changed at any time. To change the name, users have to pay a fee to the videogame company. This means that, even when changing the name is possible, users do not change their name often.

We can identify three situations for the change of name. Firstly, users want anonymity and thus, they change their name to avoid being recognized by other users. We can consider this as a "cheap" way to achieve anonymity, since it is actually cheaper than creating a new account, and the user does not have to re-classify her new account into a top-league (actually, quite expensive in terms of play-hours). Since changing the name does not require a change in the account, even with a new name users are actually easily recognizable. We will discuss this in the following section. Secondly, users may join a *team*. Teams are groups of avatars that frequently play together collaboratively against other teams. Teams also have associated names which users usually add to their names as a suffix. Thirdly, users may change their name by any other reason.

It is clear that, when finding two candidate avatars for merging, we cannot rely simply in comparing their names. On the one hand, very popular names such as "Batman" or "Superman" may be used by several users. On the other hand, even if a user has different accounts, and we successfully identify them for merging, nothing forbids the user to use different avatar names for those accounts. Thus, names will be used as *weak indicators* for avatar merging.

URL In *Starcraft 2*, as described in Table 7.3, an avatar is associated with a unique account. This account can be identified by the URL associated to the avatar which contains information about the location of the avatar (European Union, USA, Korea, etc.), the ID number and the avatar's name. As we have discussed, users are free to change the name of their avatars by paying a fee. When this is done, the URL changes by removing the old name and including the new one. However, since it is the same account, the location and the ID_number remain the same.

The URL is a *strong indicator* for avatar merging since it is quite obvious that, given two avatars

with the same associated account, they correspond to the same user. Usually, we would integrate the traces of these avatars into a single one before the trace classification step (and actually, we do this for the first of our experiments). Instead, we will leave them as they are since they provide us with a sort of "ground truth". That is, if our system is able to merge avatars of different accounts, it should be able to merge avatars of the same account.

Surrogate Avatars Given the set of traces T and the set of avatars A, we generate a partition of A in two different subsets A_γ and A_θ (a partition means that $A = A_\gamma \cup A_\theta$ and $A_\gamma \cap A_\theta = \emptyset$). For each $a \in A_\theta$, we generate a partition in T_a with components T_{a^1} and T_{a^2} where a^1, a^2 are called the surrogate avatars of a. Let $\tilde{A}_\gamma$ be the set of all surrogate avatars, we build the set $\tilde{A} = \tilde{A}_\gamma \cup A_\theta$.

Intuitively, surrogate avatars are known to belong to the same user. Thus, they provide a "ground truth" to evaluate our approach.

In Figure 7.4 the chart at left shows the long-tail distribution of the number of games played by avatar. We assume that professional players (those we are looking to disambiguate) are those that belong to the head of the curve, i.e. those that play the most. We consider this assumption fair since, in order to become professionals, players have to practice and perform in several competitions yielding a high number of games played. Thus, the set A_γ is built from a fraction γ of the avatars with the highest number of games played.

Similarly, we consider a minimal number of games (traces) to actually include the avatar a in A_θ. We assume that a user that has played a few games with an Avatar has no reasons to create a different one. We are well aware that this may not be the case for users that *already* have a different avatar and are just starting with their second. However, as we will discuss next, this induces an "imbalance" issue that may affect the classifiers' ability to group avatars of the same user. For an avatar with a few traces, the classifier will fail to provide a good prediction and the confusion matrix will present values which are explained by randomness rather than by the fact that two avatars belong to the same user. For example, consider an avatar a such as $|T_a| = 2$. Its row in $\tilde{C}^\rho$ will contain two non-zero values 0.5 in two different columns. It is easy to see that this avatar would likely conform a pattern concept with the avatars corresponding to those columns. In order to avoid this, we use a threshold θ such as for all $a \in T_\theta$ we have $|T_a| \geq \theta$.

The problem of balance refers to the fact that we are not certain about the distribution of time spent by a user among her different avatars. Put more simply, given that a user has two avatars, the question is if she prefers one over the other (meaning that one of her avatars has more traces associated than the other) or if she plays equally with one or the other (meaning that both of her traces have a similar number of traces associated). This is an important issue since it affects directly the efficacy of our approach. If in general, users play many more games with one of their avatars than the others, the classifier applied to the traces will be less effective and will tend to classify the traces of the avatars with fewer games on the avatar with more games. To study this issue in a deeper way, we introduce the parameter β as a balance between the traces distributed over the surrogate avatars. Consider that for an avatar a we have that $|T_a| = 100$, this is the avatar has 100 associated traces. When creating the surrogate users a^1 and a^2 a $\beta = 0.5$ yields that both surrogate users will have 50 associated traces, i.e. $|T_{a^1}| = |T_{a^2}| = 50$. With $\beta = 0.7$ we will have $|T_{a^1}| = 70$ and $|T_{a^2}| = 30$ and so on.

7.4.2 Evaluation Metrics

To evaluate our approach we will measure the precision, recall and f-measure of the first 100 ranked avatar clusters. Given the ranking r (after cluster score filtering using λ), we have:

$$precision(r) = \frac{TP}{TP + FP} \qquad recall(r) = \frac{TP}{TP + FN} \qquad F\text{-}measure(r) = \frac{2 \cdot precision(r) \cdot recall(r)}{precision(r) + recall(r)}$$

Where TP, FP and FN stand for true positives, false positives and false negative, respectively. We will consider true positives as combinations of avatar names ($NAMES$), URL and surrogate avatars

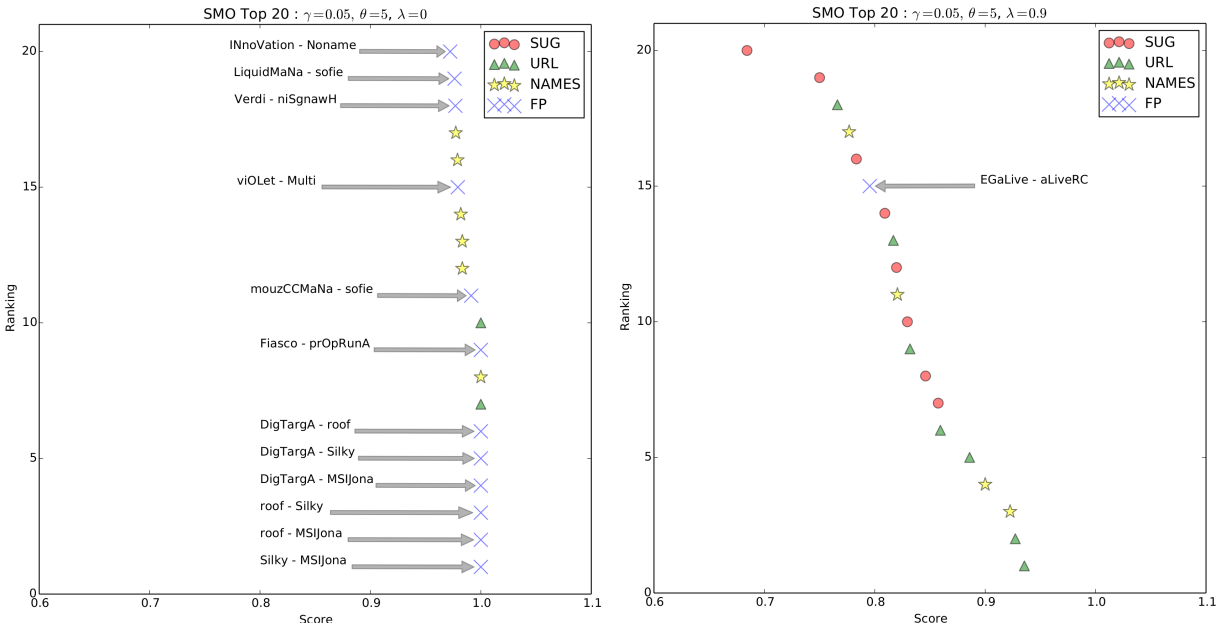


Figure 7.3: Candidate pairs ranking with $\lambda = 0$ (left) and with $\lambda = 0.9$ (right)

(*SUG*). False positives will be any candidate pair which does not belong to the true positive set in a given ranking. It is worth noticing that a pair considered as a false positive under this definition may not actually be one. We consider them false positives since we do not have enough information to consider them as true positives, meaning that their avatar names do not match, their URL is different and they are not part of our own set of surrogate avatars. *They are in fact the kind of pairs we are looking for.* False negatives are those candidate pairs that should have been considered as true positives, but that do not appear in the ranking.

The figure at left in Figure 7.3 shows the initial candidate pairs extracted from a confusion matrix generated by a Sequential Minimization Optimization (SMO) classification algorithm implemented in Weka. The classifier parameters were left as default, while the parameters of our approach for this particular figure are $\gamma = 0.05$ (top 5% of users were converted into surrogates) and $\theta = 5$ (users with less than 5 games were extracted from the dataset). Within the figure, a point represents a pair of avatars. If the avatars are surrogates, the point is represented with a red circle. If they have the same account, the point is represented with a green triangle and, in the case they have the same name, the point is a yellow star. In any other case (false positive), the point is a blue cross. False positives are annotated with the nick-names of the avatars. Figure 7.3 shows the top 20 without cluster score filtering ($\lambda = 0$) and presents very bad results. Only 8 out 20 points are not false positives (40% of precision). The figure at right in Figure 7.3 shows the top 20 after cluster score filtering ($\lambda = 0.9$) with very good results. Actually, only 1 out of 20 points is a false positive and it represents a couple of avatars that belong to the player known as `aLive`[15]. It is clear that in this particular case, our system is able to provide a precision of 100%, even though we just report 95% (19/20).

We also report on other three measures, namely P@10 (precision in the first 10 elements of the ranking), mean average precision (MAP), the receiver operating characteristic (ROC) (and the ROC area under the curve - AUC). For the sake of brevity, we do not provide a description of them. For further information on these metrics we refer the reader to [114].

[15] `http://wiki.teamliquid.net/starcraft2/ALive`

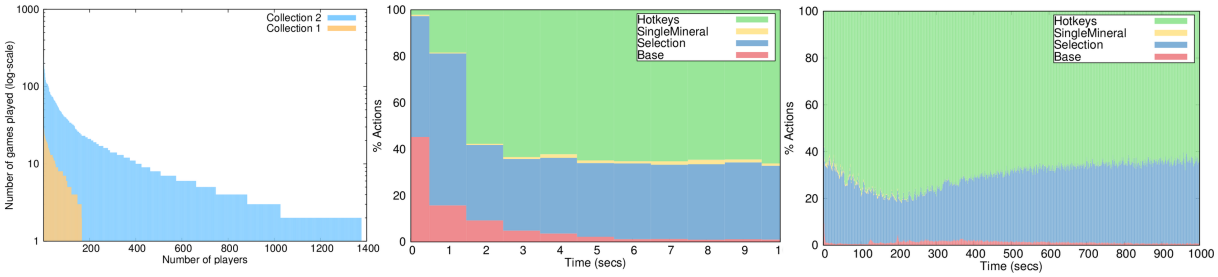


Figure 7.4: Distribution of the number of games by avatar, proportion of executed actions for first ten (resp. thousand) seconds

7.5 Experiments

This section reports a thorough evaluation of our approach, for answering the *avatar aliases identification problem*. We begin by introducing our datasets and by highlighting the prediction ability of game traces when there are no aliases in the dataset.

7.5.1 Rough replay collections

Any game of *Starcraft 2* is recorded into a file called *replay* which contains all data necessary for the game engine to replay the game. Replays are shared on dedicated websites[16]. Along with replays, a set of parsing tools which allows extracting information from the replays are openly available[17]. Using these tools we have created two collections for our study.

Collection 1 – **Replays without avatar aliases** This collection has been chosen for studying the efficacy of classifiers to recognize avatars of traces. Thus, we have purposely selected a collection of game replays which cannot contain avatar aliases. This is the case for the 2014 World Championship Series (second season[18]) in which users are forced to register their real names. The collection contains a total of 955 one-versus-one high level games and 171 unique players.

Collection 2 – **Replays with possible avatar aliases** We gathered all the replays available on the website *Spawning Tool*[19] on the month of July 2014, for a total 10,108 one-versus-one games and 3,805 players. This collection corresponds to a real world situation, and is used for evaluating our avatar alias resolution approach according to Section 7.4. Figure 7.4 shows the distribution of the number of games by avatar for both replay collections. The distribution for Collection 2 corresponds to a long tail where 10% of players participate in more than 67% of the games. The distribution for Collection 1 is explained by the elimination process of the WCS qualifiers, meaning that the distribution of game played by user gradually increments as they go up the classification ladder. Respectively, in average each player participates in 5 and 9 games in Collections 1 and 2. Charts at center and right in Figure 7.4 illustrate the proportion of each type of action used as feature for avatar classification in our approach. Particularly, these figures show the actions for the first ten and thousand seconds of the game, respectively. The object selection is the most prominent event in the first seconds of the game (the *warm up* phase), totalling 80% of the actions, after which the use of *hotkeys* becomes the most important action. This is the main reason why we will use object selection frequencies as features (in previous work, only hotkeys were used as features [170]). The chart at right in Figure 7.4 can be explained as follows. In the first minutes of the game there are not many options for the player to execute leading to a high proportion of clicking and selection events. After the 2 or 3 minutes, the user has built up a wider variety of options to execute which

[16] `http://wiki.teamliquid.net/starcraft2/Replay_Websites`
[17] http://sc2reader.readthedocs.org/
[18] http://wcs.battle.net/sc2/en/articles/wcs-2014-season-2-replays
[19] http://spawningtool.com/

leads to hotkey bindings. The major part of bindings are made only once in the game, thus the highest proportion of hotkey events around 200 seconds. In the rest of the game, these proportions stabilize.

7.5.2 Experimental set up

Two main experiments were conducted. In Collection 1, we apply a set of classifiers to test the efficacy of predicting the avatar of a trace. In Collection 2, we apply classification and clustering to perform avatar alias resolution. Both experiments share the steps of parameter selection, dataset creation and classification. The second experiment also considers a fourth step clustering, scoring and post processing.

Parameter selection Throughout this document we have described a range of parameters which compensate for the fact that we know very little of the users we are looking for. For example, the parameter γ compensates for the fact that we do not know which is the proportion of avatars in a dataset that correspond to aliases of the same user. The parameter β compensates for the fact that we do not know in which proportion aliases are used. For example, $\beta = 0.5$ represents the fact that users may use their aliases in an equal proportion, while $\beta = 0.8$ represents the fact that users may use one alias much more than the other. The parameters in our approach are a manner of discovering under which conditions the notions of avatar alias resolution holds. For this reason, we have selected a wide spectrum of parameter combinations for our experiments. The following list present the names and meaning of each parameter.

- τ: a time threshold for traces. Only the actions of the first τ seconds are considered in the dataset for classification. We also use a threshold for the number of actions, thus only the first τ actions can be considered
- γ: proportion of users converted into *surrogates aliases*
- θ: minimum number of games played by an avatar to be retained in the dataset
- β: surrogates' balance is the proportion of games attributed to one of the two surrogates derived from an avatar
- λ: cluster score threshold (see Section 7.3)

Dataset creation Having defined a set of parameter value, we generate datasets for classification from Collections 1 and 2. Each dataset contains traces as instances and avatars as class labels. Features of traces are a vector of numerical attributes and a couple of categorical attributes. Vector dimensions are associated with a canonical order over the repertory of events in the game. The value of each dimension for a given trace is the number of times the event was executed in the trace. A final dimension considers the average actions per minute (APM) associated with that trace. Categorical attributes correspond to the *race* used by the player (possible values: Protoss, Terran or Zerg) and the final status of the game (possible values: Win or Lose). Datasets are stored in the attribute-relation file format (ARFF) for the Weka system.

A single dataset is built for each different provided selection of attributes. For collection 1 we created 92 datasets, while for collection 2 we created 64 datasets.

Classification Each dataset is classified using the Weka machine learning software and evaluated using 10-fold cross validation from which we obtain a confusion matrix. To represent the generality of our approach, we chose four different classifiers, namely K Nearest Neighbours (KNN), Naive Bayes (NBAYES), J48 decision tree (J48) and Sequential Minimization Optimization (SMO). Parameters for each of the classifiers were left as default.

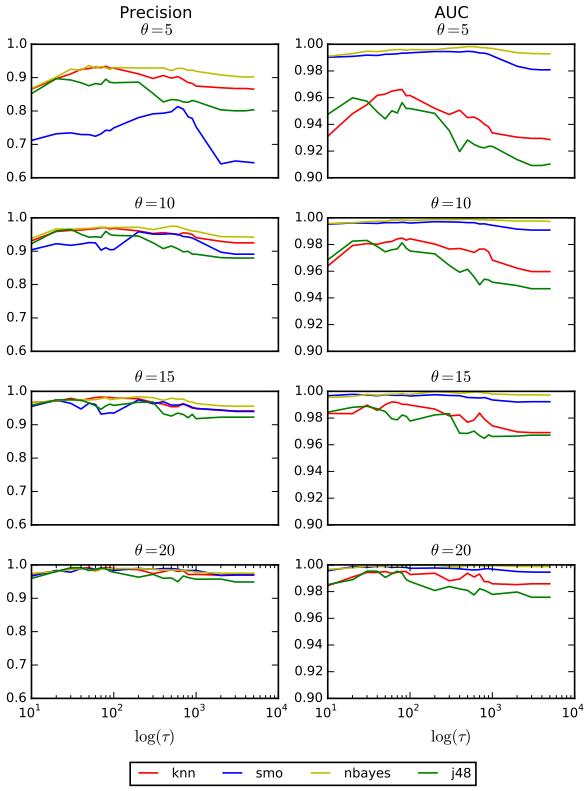


Figure 7.5: Classification results for Collection 1: precision and ROC area under the curve (AUC) distribution on 23 points of τ for four θ values. τ points were varied exponentially (10-90, 100-900, 1000-5000).

Clustering, scoring and post processing Each confusion matrix was processed by the Sephirot addIntent implementation[20] to obtain a set of pattern concepts. Scoring and post processing were implemented in ad-hoc python scripts.

7.5.3 Experimental results

Classifying avatars Figure 7.5 shows the precision and the ROC area obtained for 92 datasets created for Collection 1. The parameter τ ranged over 23 values in an exponential scale, initially from 10 to 90 seconds then from 100 to 900 and finally from 1000 to 5000 seconds (the longest game in this collection has around 5300 seconds) and thus, the x axis of each figure is in logarithmic scale. For each measure, four figures corresponding to four different settings of θ are presented. Each line corresponds to a different classifier. The figures present an empirical evaluation that the initial assumption, that avatars are very easily recognizable based in the signatures left in the traces they generate while playing, is true. For each different setting, ROC area is always around 100% showing the robustness of the approach under different parametrizations. Precision is always maintained over 60%, achieving it minimal value for the SMO classifier with $\theta = 5$ and $\tau > 1000$. Actually, this also confirms two of our previous assumptions. Firstly, it is hard to recognize users that have played a few games, meaning that the larger the value of the θ threshold, more discriminative power has the classifier. Secondly, users are recognizable in the first few minutes of the game. The precision curves show a slight concave behaviour hinting a maximum of the precision w.r.t. the time cut used for traces. Indeed, this agrees with the hotkey use description given for Figure 7.4 in the chart at right. Users can be efficiently discovered by their hotkeys binding settings. As the game progress, traces may differ given that the number of options in the game greatly increase

[20] https://code.google.com/p/sephirot/

and vary in execution regarding different opponents.

Identifying multiple aliases The goal of experiment 2 is evaluating our approach for finding avatar aliases. We have generated 48 datasets considering different parametrizations. As already discussed in the previous experiment, the efficacy of the classifiers achieves its best in the first few minutes of the game. Thus, we have selected three different τ values, namely $30, 60$ and 90 seconds. We have picked the same values for θ as in the previous experiments. Surrogates were generated for the first 5, 10, 15 and 20 percent of the most active users in the dataset (γ). For this particular experiment, we have set the balance $\beta = 0.5$. All 48 datasets were processed using the four classifiers previously mentioned yielding a total of 192 confusion matrices. The top part of the Table 7.4 shows a summary for the evaluation results using the top 100 pairs of avatar clusters found with parameters $\tau = 90$, $\theta = 20$ (248 avatars including surrogates), $\gamma = 0.2$ (41 surrogates) and $\lambda = 0.9$. The top, medium and bottom parts of the table contain the evaluations when looking for surrogates only, surragates and URLs, surrogates, URLs and names, respectively.

Results indicate that our approach is very efficient at identifying surrogate avatars with these parameters. This is particularly true for KNN and the J48 classifiers achieving very high recall values. In the upper table, while precision is low it is worth noticing that in the top 100, there are only 41 surrogates meaning that the maximum achievable precision is 0.41. The classifier KNN is particularly good in this measure achieving an almost perfect value (0.4 of 0.41). All four classifiers achieve a very high precision in the first 10 results (P@10) while two of them get a perfect score. Indeed, one of the main characteristics of our approach is the good ranking it generates over the avatar pairs. This fact is confirmed by the good MAP and ROC area under the curve (AUC) values achieved by all four classifiers. Both these measures slightly degrade when including in the set of true positives URLs and names. In the case of the latter, this can be understood since not all avatars with the same name necessarily belong to the same user (as we have previously stated, same nick-names is a weak indicator). Thus, pairs of avatars with the same name will be more evenly distributed over the ranking or can even be found at the bottom indicating that they do not belong to the same user. This fact is reflected in the gap between the high grow of precision and low degradation of recall, i.e. avatars with the same name are evenly distribute between the pairs retrieved and those that were not.

A special mention deserve the URL true positives. As we have discussed, avatars with the same URL necessarily belong to the same user. Hence, we would have expected that in the first 10 pairs retrieved we could find an even distribution of surrogates and URLs. Instead, for all classifiers, P@10 is more than 80% surrogates (while the rest is always URLs - P@10 in the medium part of the table). The reason behind this is that we have purposely selected a balance of 0.5 for the surrogate distribution of traces, while we do not have control over this value for the URL pairs. The lower part of Table 7.4 shows a summary of results when looking for just surrogates while varying the balance in the distribution of traces between them. We can clearly observe that the performance of the approach quickly degrades as more imbalanced gets the distribution (the higher the β value). Actually, for some classifiers it is not possible to obtain a single good result, even when we have lowered the λ threshold to 0.8. As URLs are not necessarily balanced, classifiers tend to predict the label of a trace belonging to an avatar with less traces to one with more traces. Issues related to learning from imbalanced datasets are reviewed in [77] and need to be considered when selecting a proper classifier for our particular application.

7.6 Related Work

Recently, Yan et. al [170] suggested that behavioural patterns discriminating skills, but also player themselves, can be discovered in the way they use their keyboard when playing *Starcraft 2*. They gathered 3,316 replays and took as features the frequencies of each control group key were used (30 features) for the whole game. They showed that these features allow predicting with high accuracy the

Parameters:: $\gamma = 0.2$, $\theta = 20$, $\lambda = 0.9$, $\tau = 90$						
Surrogates						
Classifier	F1	MAP	Recall	AUC	Precision	P@10
j48	0.468	0.824	0.805	0.904	0.33	1.0
naivebayes	0.226	0.740	0.390	0.915	0.16	0.8
smo	0.312	0.971	0.536	0.993	0.22	1.0
knn	0.567	0.822	0.976	0.882	0.4	0.9
Surrogates & URLS						
Classifier	F1	MAP	Recall	AUC	Precision	P@10
j48	0.588	0.907	0.606	0.866	0.57	1.0
naivebayes	0.443	0.857	0.457	0.864	0.43	1.0
smo	0.257	0.912	0.266	0.945	0.25	1.0
knn	0.670	0.937	0.691	0.874	0.65	1.0
Surrogates & URLS & Names						
Classifier	F1	MAP	Recall	AUC	Precision	P@10
j48	0.689	0.983	0.606	0.935	0.8	1.0
naivebayes	0.560	0.943	0.492	0.906	0.65	1.0
smo	0.258	0.949	0.227	0.960	0.3	1.0
knn	0.758	0.967	0.667	0.792	0.88	1.0
Parameters:: $\gamma = 0.2$, $\theta = 20, \lambda = 0.8$, $\tau = 90$						
J48						
Balance	F1	MAP	Recall	AUC	Precision	P@10
$\beta = 0.5$	0.925	0.996	0.929	0.955	0.920	1.0
$\beta = 0.6$	0.545	0.927	0.632	0.921	0.480	1.0
$\beta = 0.7$	0.053	0.695	0.077	0.977	0.040	0.3
$\beta = 0.8$	-	-	-	-	-	-
Naive Bayes						
Balance	F1	MAP	Recall	AUC	Precision	P@10
$\beta = 0.5$	0.472	0.902	0.475	0.953	0.470	0.9
$\beta = 0.6$	0.273	0.923	0.316	0.973	0.240	1.0
$\beta = 0.7$	0.197	0.9	0.288	0.978	0.150	0.9
$\beta = 0.8$	0.048	0.533	0.120	0.983	0.030	0.3
SMO						
Balance	F1	MAP	Recall	AUC	Precision	P@10
$\beta = 0.5$	0.392	0.983	0.394	0.992	0.390	1.0
KNN						
Balance	F1	MAP	Recall	AUC	Precision	P@10
$\beta = 0.5$	0.905	0.964	0.909	0.732	0.9	1.0
$\beta = 0.6$	0.750	0.957	0.868	0.929	0.660	1.0
$\beta = 0.7$	0.184	0.706	0.269	0.949	0.140	0.7

Table 7.4: Summary of evaluation measures over the resulting avatar cluster list yielded by the alias resolution approach (at top), and avatar clustering when varying the balance (β) (at bottom). Each entry represents a confidence matrix yielded by the respective classifier

league in which an avatar is playing[21]. A second result tells that a basic SVM classifier can predict the avatars involved in a game with high accuracy (≥ 0.95 accuracy with a leave-one-out validation), even when the avatar has few numbers of samples (between 2 and 20). Yan et. al showed that *hotkeys* (control groups) yield unique behavioural patterns of a user, but they did not present a way to discover avatar aliases: they even removed avatars with high probability of being aliases (e.g., *bar code names).*

Using control groups as features is actually inspired by several works in software and security applications. Indeed, typing patterns allow identifying users by their typing characteristics [132]. The tedious task is to determine the appropriate behavioural metrics and features [169]. For example, keystroke

[21] Leagues regroup players by level, following an ELO-like ranking system, from bronze, silver, gold, platinum, diamond, master, to grand master, the later involving the best 200 players of each continent.

dynamics and typing rhythm are crucial for authenticating a user based on habitual patterns [123]. Recently, an investigation around *Starcraft 2* [154] highlighted that the predictive importance of features is not constant across levels of expertise, while Yan et al. [170] ensured that complex features, even spatio-temporal, are not important: only the frequency of *hotkeys* is enough to output highly accurate classifiers; we emphasized this fact by showing that only the first few minutes of game play are enough to recognize the player.

Starcraft 2 and other real time strategy games (RTS) in general, face several research challenges in artificial intelligence [128] including *opponent modelling and learning*. Game traces/logs from replay files tend to be more and more used to tackle these problems since this information is easy and free to gather. We can notice several works focusing especially on tactical and strategic aspects, such as predicting army locations and opponents actions [164, 151] and automatically discovering build orders [109, 32]. All these works focus on effective actions made by players (build orders, micro/macro management) while we use here only the very first few actions of the *warm up* phase that one could consider as noise.

7.7 Conclusion

Video game analytics is a growing field of data science, crucial, if not vital, for the biggest game producers and editors. It comes with many challenges, the holy grail being to find all the ingredients that could assure an indisputable success of a game directly at its release. Pragmatically however, behavioral *big data* is gathered and analyzed to answer several problems, including the design of better artificial agents, game balancing, bugs and cheaters and usurpers detection, etc. Games are then patched, cheaters are banned, and this cycle restarts. Behavioral data is also a gold mine in the context of electronic sports and competitive gaming, for reaching the same goals as in standard *sport analytics* (see e.g. `http://www.sloansportsconference.com`).

We introduced the problem of avatar aliases identification, when there exists no mapping between individuals and their avatars. This is an important problem for game editors, but also for e-sport structures. Our method relies on the fact that behavioral data hide individual characteristic patterns, which allows making predictive approaches very accurate. Nevertheless, this good performance quickly degrades when data hides avatar aliases, which is why we based our analysis on confusion matrices.

Going further, we proposed an original method that considers the lattice of binary classifiers, where each element is a model learned from positive and negative examples that are respectively the instances of a subset of labels B and their complementary instances. This constitutes the search space of our problem and each binary classifier forms a potential group of avatar duplicates. We propose an efficient way to traverse the lattice of binary classifier to output the set of *duplicate label sets* [101].

Conclusion

There are many ways to write a manuscript for an "Habilitation". I chose to give, as an introduction, an overview of the research problems that attracted me over the last ten years before developing on the most representative contributions covering the different axes. It is important to notice that I did not always chose beforehand the problems to investigate, but the applications from collaborative project opportunities did it for us. This is probably the most interesting aspect of research in KDD: the actual data and the experimental results can whisper real challenges. Going back to theory helps formalizing the problem, connecting it with other problems, and sharing with other researchers in different communities. The most representative case was the problem of neuroscience where the objective was to extract descriptive rules between molecular descriptors and odors. Subgroup discovery was already a widely studied task in data mining, but the state of the art at that moment was presenting several challenging points according to the dataset we had in hands.

PhD and post-doctoral researchers supervision also strongly supports the realization of the research. Their propositions inevitably impacts the research line, as well as their wishes to focus on particular problems. I shall name and specially thank them again. Guillaume Bosc (2014-2017) was attracted in algorithms for subgroup discovery and making sense of data. Olivier Cavadenti (2013-2016) was more attracted by methodological aspects of KDD by (non-trivially) bringing existing pieces together to answer a company need through a collaboration. Aimene Belfodil (2016-2019) literally fell in love with Formal Concept Analysis and focused on studying its limits and open challenges for manipulating non standard patterns. Finally, Romain Mathonat (2017-), got interested in sequential pattern mining with non exhaustive techniques for mining video game data logs. Víctor Codocedo, during his post-doctoral research (2015-2016), worked on video game analytics for the application side and also on sequential patterns characterization with FCA and mining. Allowing researchers under supervision to work on what they enjoy, still ensuring that their scope fits an open question within a coherent and long term research line was actually a role I had to endorse, and is, in my honest opinion, the most important requirement for an "Habilitation". The other collaborations, within the team, or with other research laboratories, are also mandatory, in order to stay aware, to confront ideas and opinions in a gentle atmosphere and build on the different singularities of each other.

Collaborations with industries and other researchers –but not computer scientists– has also influenced my research (and career, as I am currently in a long term sabbatical of my associate professor position). Being involved in long term national or European projects, with an important financial support for the different partners forces to concentrate not only on the theory and methods, but on the actionability of the produced results: *What will stay once the project is over*? Such projects are not "pure research" supports but integrate a "knowledge and method transfer" component which was at the beginning disturbing for me. Indeed, it took us several years between the very first discussion with neuroscientists (especially Dr. Moustafa Bensafi) and the publication of actionable results in a specialized journal. It will take some time before that our tools for eliciting hypotheses from olfactory data will be used as a routine in other laboratories (and it may never happen without a serious follow up).

Such remark can be made also for the video game analytics application. In that particular case, I mostly played the role of domain expert for StarCraft II, but here again, producing interesting patterns took us quite some time between the first ideas, my stay at the MIT Media Lab (with my colleague and friend Chedy Raïssy) and the final publication. Interestingly, manipulating and understanding those data

(along with social media data and a few others) helped me to propose practical sessions for students in our school (INSA Lyon) for which I knew precisely what to discover within (knowledge, models, errors, artifacts, etc.). It consequently helped me to prepare these data mining and machine learning classes, attract students and communicate them the main messages. Actually, the transfer is at its preliminary stage at the university: engineers who will leave the school have experimented KDD with real problems (yet simplified for the purpose of the class) and may remember that pattern discovery can be useful at some point in their career.

Going back to my research project, I described in this manuscript three major axes: (i) Theoretical aspects of pattern discovery with Formal Concept Analysis as a bridge to Order Theory; (ii) Subgroup Discovery: interestingness and algorithms, and finally (iii) applications for practical and tangible validation and transfer. My perspectives of research still fit in these three axes, with slight amendments.

(i) Data and Pattern Formalization This axis of research started during my PhD. The original problem was to consider an exact numerical pattern enumeration and FCA played a key role. It was possible to very elegantly define and mine such patterns, simply by defining what an intersection is. This is the beauty of *pattern structures* but also its weakness. The beauty lies in the fact that it allowed us to considered many pattern types very simply (biclusters, functional dependencies, intervals, polygons, etc.) and heterogeneous data on those types. The weakness is that there is not always a "unique intersection" for some very interesting types of patterns (sequences, graphs, circles, ...). For many years, this is what reduced the visibility of pattern structures. This is why we developed on pattern setups and pattern multi-structures. We are only at the beginning of manipulating and understanding such structures.

(ii) Subgroup Discovery and Algorithms Although we mainly defined Subgroup Discovery as the task of finding patterns that discriminate a class, the original definition of Wrobel is more imprecise and actually covers exceptional model mining. Nevertheless, we were attracted by subgroup discovery as soon as basic constraints of constrained pattern mining were insufficient (or simply hazardous or impossible to set, as choosing the right combination of constraints becomes a pattern mining problem itself!). Our perspectives in this axis concern algorithms and interestingess. Concerning the algorithms, we are investigating how to generically sample (minimal generators of) extents (object sets), that is, proposing bandit based methods for any pattern domain (provided or not with a memory as it is the case for MCTS). We hope to add guarantees systematically for anytime algorithms that tell how fare we are from an exhaustive search, if the best pattern has been found or not, etc. Following the idea of mining finer and finer discretizations, our goal is to formalize a partial order of pattern domains (pattern structure projections) and start the exploration in the most general representations. Then, when a subspace is promising, finer data representations can be considered. That was actually the initial subject of Guillaume Bosc's PhD thesis in 2014, and we are making progress slowly but surely. The other aspects that we will start to investigate is the inclusion of the domain expert during the search, by learning its preferences and guiding the search. Considering anytime approaches with an exploration/exploitation trade-off, we can imagine the expert to give feedback on some result snapshots, which can be then handled by the exploration/exploitation strategy.

(iii) KDD in practice To stay close of real problems, contact with the real world shall be kept. I decided to make a big move in that direction, by taking a long term sabbatical in a company. There, the main problems are linked to its important and constant growth over the past decades (from a few to 250 employees). It comes with many issues which can be gathered under the term of "Digitization" or "Numeric transformation", many of which than can be approached thanks to the following tasks.

Static Code and Software Analytics To understand and optimize processes and information retrieval, many sources of data can be valorized, but should be first available, and before that, modeled. For that matter, I got attracted by Software Analytics through the construction of the Abstract Syntax Tree (AST) of a 19 millions lines code, in which many entities and links

between those entities are hidden and required for having a precise information system on the company activities and processes. Pattern mining techniques are adapted for many tasks.

Predictive Maintenance Another field that I discovered through this experience is predictive/preventive maintenance. Our software runs on industrial environments and we can gather many types of monitoring time series and events from the software environment (Java Virtual Machine, the machine itself and the database system). We observed that several problems could have been easily anticipated while others can be discovered with pattern mining.

Knowledge Spaces Over the last years, public institutions and private companies are increasingly in the need of understanding the knowledge state of their group to answer different challenges such as efficiently *training/learning* for the employee side, and *identifying critical skills* for the employer side. These challenges naturally arise in the academic system: universities have to define skills students can learn (and dependencies between skills), and to regularly assess the knowledge state of a student and a class with exams, and most importantly, use the knowledge state to personalize the evaluations and learning trajectories. For this matter, a theory has been successfully developed several years ago and is now successfully used by millions of students, under an implementation known as ALEKS (`https://www.aleks.com/about_aleks/knowledge_space_theory`). This theory is actually strongly rooted in FCA [61]. We propose to go beyond the current state of research of *knowledge spaces*. Indeed, the theory is centered on a single user, while we need to understand the knowledge state of a group. We are currently building a knowledge space under elaboration, and ways to automatically assess skills through user traces at a frequency rate that will allow realistic experimentation.

Behavioral Data Analytics Finally, we will attach a particular interest in analyzing user and system traces for several purposes (other than for assessing the knowledge state of a user) such as detecting anomalies in the usage of our software and detecting security issues. The variety of data sources and the volume of data makes there real challenges from which I am convinced to discover research problems.

Bibliography

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, Reading (MA), USA, 1995.

[2] B. Abramson. Expected-outcome: A general model of static evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(2):182–193, 1990.

[3] T. Abudawood and P. A. Flach. Evaluation measures for multi-class subgroup discovery. In *ECML PKDD*, pages 35–50. Springer, 2009.

[4] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734 – 749, june 2005.

[5] C. C. Aggarwal. *Data Mining - The Textbook*. Springer, 2015.

[6] C. C. Aggarwal and J. Han, editors. *Frequent Pattern Mining*. Springer, 2014.

[7] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data. *Data Min. Knowl. Discov.*, 11(1):5–33, 2005.

[8] 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, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216. ACM Press, 1993.

[9] D. W. Aha, M. Molineaux, and M. J. V. Ponsen. Learning to win: Case-based plan selection in a real-time strategy game. In *Case-Based Reasoning, Research and Development, 6th International Conference, on Case-Based Reasoning, ICCBR 2005*, pages 5–20, 2005.

[10] M. A. Ahmad, B. Keegan, J. Srivastava, D. Williams, and N. S. Contractor. Mining for gold farmers: Automatic detection of deviant players in mmogs. In *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering, CSE 2009*, pages 340–345, 2009.

[11] F. Alqadah and R. Bhatnagar. Similarity measures in formal concept analysis. *Ann. Math. Artif. Intell.*, 61(3):245–256, 2011.

[12] S. Andrews. A 'Best-of-Breed' approach for designing a fast algorithm for computing fixpoints of Galois Connections. *Inf. Sci.*, 295:633–649, 2015.

[13] K. Arulkumaran, A. Cully, and J. Togelius. Alphastar: An evolutionary computation perspective. *CoRR*, abs/1902.01724, 2019.

[14] M. Atzmüller and F. Lemmerich. Fast subgroup discovery for continuous target concepts. In J. Rauch, Z. W. Ras, P. Berka, and T. Elomaa, editors, *Foundations of Intelligent Systems, 18th International Symposium, ISMIS 2009, Prague, Czech Republic, September 14-17, 2009. Proceedings*, volume 5722 of *Lecture Notes in Computer Science*, pages 35–44. Springer, 2009.

[15] M. Atzmüller and F. Puppe. Sd-map - A fast algorithm for exhaustive subgroup discovery. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4213 of *Lecture Notes in Computer Science*, pages 6–17. Springer, 2006.

[16] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[17] J. Baixeries. *Lattice Characterization of Armstrong and Symmetric Dependencies (PhD Thesis)*. Universitat Politècnica de Catalunya, 2007.

[18] J. Baixeries, V. Codocedo, M. Kaytoue, and A. Napoli. Characterizing approximate-matching dependencies in formal concept analysis with pattern structures. *Discrete Applied Mathematics*, 249:18–27, 2018.

[19] J. Baixeries, M. Kaytoue, and A. Napoli. Computing similarity dependencies with pattern structures. In M. Ojeda-Aciego and J. Outrata, editors, *Proceedings of the Tenth International Conference on Concept Lattices and Their Applications, La Rochelle, France, October 15-18, 2013.*, volume 1062 of *CEUR Workshop Proceedings*, pages 33–44. CEUR-WS.org, 2013.

[20] J. Baixeries, M. Kaytoue, and A. Napoli. Characterizing functional dependencies in formal concept analysis with pattern structures. *Ann. Math. Artif. Intell.*, 72(1-2):129–149, 2014.

[21] S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets. *Data Min. Knowl. Discov.*, 5(3):213–246, 2001.

[22] A. Belfodil, S. O. Kuznetsov, C. Robardet, and M. Kaytoue. Mining convex polygon patterns with formal concept analysis. In *IJCAI 2017*, pages 1425–1432, 2017.

[23] A. A. Bendimerad, M. Plantevit, and C. Robardet. Unsupervised exceptional attributed sub-graph mining in urban data. In F. Bonchi, J. Domingo-Ferrer, R. A. Baeza-Yates, Z. Zhou, and X. Wu, editors, *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 21–30. IEEE, 2016.

[24] J. Besson, C. Robardet, and J.-F. Boulicaut. Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In H. Schärfe, P. Hitzler, and P. hrstrøm, editors, *Conceptual Structures: Inspiration and Application, 14th International Conference on Conceptual Structures (ICCS)*, volume 4068 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 2006.

[25] J. Besson, C. Robardet, L. D. Raedt, and J.-F. Boulicaut. Mining bi-sets in numerical data. In S. Dzeroski and J. Struyf, editors, *KDID*, volume 4747 of *Lecture Notes in Computer Science*, pages 11–23. Springer, 2007.

[26] Y. Björnsson and H. Finnsson. Cadiaplayer: A simulation-based general game player. *IEEE Trans. Comput. Intellig. and AI in Games*, 1(1):4–15, 2009.

[27] 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.

[28] M. Boley, C. Lucchese, D. Paurat, and T. Gärtner. Direct local pattern sampling by efficient two-step random procedures. In C. Apté, J. Ghosh, and P. Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 582–590. ACM, 2011.

[29] M. Boley, S. Moens, and T. Gärtner. Linear space direct pattern sampling using coupling from the past. In *KDD*, pages 69–77, 2012.

[30] G. Bosc, J.-F. Boulicaut, C. Raïssi, and M. Kaytoue. Anytime discovery of a diverse set of patterns with monte carlo tree search. *Data Min. Knowl. Discov.*, 32(3):604–650, May 2018.

[31] G. Bosc, J. Golebiowski, M. Bensafi, C. Robardet, M. Plantevit, J. Boulicaut, and M. Kaytoue. Local subgroup discovery for eliciting and understanding new structure-odor relationships. In T. Calders, M. Ceci, and D. Malerba, editors, *Discovery Science - 19th International Conference, DS 2016, Bari, Italy, October 19-21, 2016, Proceedings*, volume 9956 of *Lecture Notes in Computer Science*, pages 19–34, 2016.

[32] G. Bosc, P. Tan, J.-F. Boulicaut, C. Raïssi, and M. Kaytoue. A Pattern Mining Approach to Study Strategy Balance in RTS Games. *IEEE Trans. Comput. Intellig. and AI in Games*, 9(2):123–132, June 2017.

[33] J. Boulicaut and B. Jeudy. Constraint-based data mining. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 339–354. Springer, 2010.

[34] R. Braga Araújo, G. Trielli Ferreira, G. Orair, J. Meira, Wagner, R. Celso Ferreira, D. Olavo Guedes Neto, and M. Zaki. The partricluster algorithm for gene expression analysis. *International Journal of Parallel Programming*, 36:226–249, 2008.

[35] B. Bringmann and A. Zimmermann. One in a million: picking the right patterns. *Knowl. Inf. Syst.*, 18(1):61–81, 2009.

[36] C. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. P. Liebana, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intellig. and AI in Games*, 4(1):1–43, 2012.

[37] A. Buzmakov, S. O. Kuznetsov, and A. Napoli. Fast generation of best interval patterns for nonmonotonic constraints. In *ECML PKDD*, pages 157–172. Springer, 2015.

[38] A. Buzmakov, S. O. Kuznetsov, and A. Napoli. Revisiting pattern structure projections. In *Formal Concept Analysis*, pages 200–215. Springer, 2015.

[39] A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 75–85. AAAI, 2000.

[40] C. J. Carmona, P. González, M. J. del Jesús, and F. Herrera. NMEEF-SD: non-dominated multiobjective evolutionary algorithm for extracting fuzzy rules in subgroup discovery. *IEEE Trans. Fuzzy Systems*, 18(5):958–970, 2010.

[41] L. Caruccio, V. Deufemia, and G. Polese. Relaxed functional dependencies - A survey of approaches. *IEEE Trans. Knowl. Data Eng.*, 28(1):147–165, 2016.

[42] N. Caspard and B. Monjardet. The lattices of closure systems, closure operators, and implicational systems on a finite set: A survey. *Discrete Applied Mathematics*, 127(2):241–269, 2003.

[43] L. Cerf, J. Besson, C. Robardet, and J.-F. Boulicaut. Closed patterns meet n-ary relations. *TKDD*, 3(1), 2009.

[44] 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.

[45] G. Cheung and J. Huang. Starcraft from the stands: understanding the game spectator. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 763–772. ACM, 2011.

[46] V. Codocedo, J. Baixeries, M. Kaytoue, and A. Napoli. Characterization of order-like dependencies with formal concept analysis. In M. Huchard and S. Kuznetsov, editors, *Proceedings of the Thirteenth International Conference on Concept Lattices and Their Applications, Moscow, Russia, July 18-22, 2016.*, volume 1624 of *CEUR Workshop Proceedings*, pages 123–134. CEUR-WS.org, 2016.

[47] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK, 1990.

[48] M. J. del Jesús, P. González, F. Herrera, and M. Mesonero. Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. *IEEE Trans. Fuzzy Systems*, 15(4):578–592, 2007.

[49] K. Denecke and S. L. Wismath. Galois connections and complete sublattices. In *Galois Connections and Applications*, pages 211–229. Springer, 2004.

[50] E. W. Dereszynski, J. Hostetler, A. Fern, T. G. Dietterich, T. Hoang, and M. Udarbe. Learning probabilistic behavior models in real-time strategy games. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2011*, pages 20–25. The AAAI Press, 2011.

[51] O. Devillers. On Deletion in Delaunay Triangulations. In V. Milenkovic, editor, *Proceedings of the Fifteenth Annual Symposium on Computational Geometry, Miami Beach, Florida, USA, June 13-16, 1999*, pages 181–188. ACM, 1999.

[52] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In U. M. Fayyad, S. Chaudhuri, and D. Madigan, editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999*, pages 43–52. ACM, 1999.

[53] W. Duivesteijn, A. Feelders, and A. J. Knobbe. Exceptional model mining - supervised descriptive local pattern mining with complex target concepts. *Data Min. Knowl. Discov.*, 30(1):47–98, 2016.

[54] W. Duivesteijn, A. J. Knobbe, A. Feelders, and M. van Leeuwen. Subgroup discovery meets bayesian networks – an exceptional model mining approach. In G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, editors, *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 158–167. IEEE Computer Society, 2010.

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

[56] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Trans. Knowl. Data Eng.*, 23(5):683–698, 2011.

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

[58] 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.

[59] 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.

[60] J. Fürnkranz, D. Gamberger, and N. Lavrac. *Foundations of Rule Learning*. Cognitive Technologies. Springer, 2012.

[61] B. Ganter, M. Bedek, J. Heller, and R. Suck. An invitation to knowledge space theory. In *Formal Concept Analysis - 14th International Conference, ICFCA 2017, Proceedings*, pages 3–19, 2017.

[62] B. Ganter and S. O. Kuznetsov. Pattern structures and their projections. In *ICCS 2001*, LNCS (2120). 129–142., 2001.

[63] B. Ganter and S. Obiedkov. *Conceptual Exploration*. Springer, Berlin, 2016.

[64] B. Ganter and R. Wille. *Formal Concept Analysis*. Springer, 1999.

[65] M. García-Borroto, J. F. M. Trinidad, and J. A. Carrasco-Ochoa. A survey of emerging patterns for supervised classification. *Artif. Intell. Rev.*, 42(4):705–721, 2014.

[66] G. C. Garriga, P. Kralj, and N. Lavrac. Closed sets for labeled data. *Journal of Machine Learning Research*, 9:559–580, 2008.

[67] R. Gaudel and M. Sebag. Feature selection as a one-player game. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 359–366. Omnipress, 2010.

[68] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9, 2006.

[69] A. Giacometti, D. H. Li, P. Marcel, and A. Soulet. 20 years of pattern mining: a bibliometric survey. *SIGKDD Explor. Newsl.*, 15(1):41–50, 2013.

[70] A. Giacometti and A. Soulet. Dense neighborhood pattern sampling in numerical data. In *SIAM*, pages 756–764, 2018.

[71] C. V. Glodeanu, M. Kaytoue, and C. Sacarea, editors. *Formal Concept Analysis - 12th International Conference, ICFCA 2014, Cluj-Napoca, Romania, June 10-13, 2014. Proceedings*, volume 8478 of *Lecture Notes in Computer Science*. Springer, 2014.

[72] G. Graetzer, B. Davey, R. Freese, B. Ganter, M. Greferath, P. Jipsen, H. Priestley, H. Rose, E. Schmidt, S. Schmidt, F. Wehrung, and R. Wille. *General Lattice Theory*. Freeman, San Francisco, CA, 1971.

[73] H. Grosskreutz and S. Rüping. On subgroup discovery in numerical domains. *Data Min. Knowl. Discov.*, 19(2):210–226, 2009.

[74] J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95:5–18, 1986.

[75] T. Guyet, R. Quiniou, and V. Masson. Mining relevant interval rules. *CoRR*, abs/1709.03267, 2017.

[76] J. A. Hartigan. Direct Clustering of a Data Matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

[77] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 2009.

[78] D. P. Helmbold and A. Parker-Wood. All-moves-as-first heuristics in monte-carlo go. In H. R. Arabnia, D. de la Fuente, and J. A. Olivas, editors, *Proceedings of the 2009 International Conference on Artificial Intelligence, ICAI 2009, July 13-16, 2009, Las Vegas Nevada, USA, 2 Volumes*, pages 605–610. CSREA Press, 2009.

[79] F. Herrera, C. J. Carmona, P. González, and M. J. del Jesús. An overview on subgroup discovery: foundations and applications. *Knowl. Inf. Syst.*, 29(3):495–525, 2011.

[80] Q. Hu and T. Imielinski. Alpine: Progressive itemset mining with definite guarantees. In *SIAM*, pages 63–71, 2017.

[81] D. P. Huttenlocher, G. A. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):850–863, 1993.

[82] D. I. Ignatov, S. O. Kuznetsov, and J. Poelmans. Concept-based biclustering for internet advertisement. In J. Vreeken, C. Ling, M. J. Zaki, A. Siebes, J. X. Yu, B. Goethals, G. I. Webb, and X. Wu, editors, *12th IEEE International Conference on Data Mining Workshops (ICDM)*, pages 123–130. IEEE Computer Society, 2012.

[83] R. Jäschke, A. Hotho, C. Schmitz, B. Ganter, and G. Stumme. Trias - an algorithm for mining iceberg tri-lattices. In *ICDM*, pages 907–911, 2006.

[84] L. Ji, K.-L. Tan, and A. K. H. Tung. Mining frequent closed cubes in 3d datasets. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, pages 811–822. ACM, 2006.

[85] P. C. Kanellakis. Elements of relational database theory. In J. van Leeuwen, editor, *Handbook of theoretical computer science (vol. B)*, pages 1073–1156. MIT Press, Cambridge, MA, USA, 1990.

[86] M. Kaytoue, Z. Assaghir, A. Napoli, and S. O. Kuznetsov. Embedding tolerance relations in formal concept analysis: an application in information fusion. In *CIKM*, pages 1689–1692. ACM, 2010.

[87] M. Kaytoue, V. Codocedo, J. Baixeries, and A. Napoli. Three interrelated FCA methods for mining biclusters of similar values on columns. In K. Bertet and S. Rudolph, editors, *Proceedings of the Eleventh International Conference on Concept Lattices and Their Applications, Košice, Slovakia, October 7-10, 2014.*, volume 1252 of *CEUR Workshop Proceedings*, pages 243–254. CEUR-WS.org, 2014.

[88] M. Kaytoue, S. O. Kuznetsov, J. Macko, and A. Napoli. Biclustering meets triadic concept analysis. *Ann. Math. Artif. Intell.*, 70(1-2):55–79, 2014.

[89] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Biclustering numerical data in formal concept analysis. In P. Valtchev and R. Jäschke, editors, *ICFCA*, volume 6628 of *LNCS*, pages 135–150. Springer, 2011.

[90] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Revisiting Numerical Pattern Mining with Formal Concept Analysis. In *IJCAI*, pages 1342–1347, 2011.

[91] M. Kaytoue, S. O. Kuznetsov, A. Napoli, and S. Duplessis. Mining gene expression data with pattern structures in fca. *Inf. Sci.*, 181(10):1989–2001, 2011.

[92] M. Kaytoue, M. Plantevit, A. Zimmermann, A. A. Bendimerad, and C. Robardet. Exceptional contextual subgraph mining. *Machine Learning*, 106(8):1171–1211, 2017.

[93] M. Kaytoue, A. Silva, L. Cerf, W. M. Jr., and C. Raïssi. Watch me playing, i am a professional: a first study on video game live streaming. In A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, editors, *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 1181–1188. ACM, 2012.

[94] M. Kaytoue-Uberall, S. Duplessis, S. O. Kuznetsov, and A. Napoli. Two fca-based methods for mining gene expression data. In S. Ferré and S. Rudolph, editors, *Proceedings of the 7th International Conference on Formal Concept Analysis (ICFCA)*, volume 5548 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2009.

[95] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer, 2006.

[96] L. Kurgan and K. J. Cios. Discretization algorithm that uses class-attribute interdependence maximization. In *IC-AI*, pages 980–987, 2001.

[97] S. O. 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.

[98] 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, *Formal Concept Analysis, Foundations and Applications*, Lecture Notes in Computer Science 3626, pages 196–225. Springer, 2005.

[99] 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.

[100] S. O. Kuznetsov and J. Poelmans. Knowledge representation and processing with formal concept analysis. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 3(3):200–215, 2013.

[101] Q. Labernia, V. Codocedo, C. Robardet, and M. Kaytoue. Mining the lattice of binary classifiers for identifying duplicate labels in behavioral data. In S. Benferhat, K. Tabia, and M. Ali, editors, *Advances in Artificial Intelligence: From Theory to Practice - 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, Arras, France, June 27-30, 2017, Proceedings, Part II*, volume 10351 of *Lecture Notes in Computer Science*, pages 12–21. Springer, 2017.

[102] N. Lavrac, P. A. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In S. Dzeroski and P. A. Flach, editors, *Inductive Logic Programming, 9th International Workshop, ILP-99, Bled, Slovenia, June 24-27, 1999, Proceedings*, volume 1634 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 1999.

[103] M. Leece and A. Jhala. Sequential pattern mining in starcraft: Brood war for short and long-term goals. In *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2014*, pages 281–288. The AAAI Press, 2014.

[104] F. Lehmann and R. Wille. A triadic approach to formal concept analysis. In *ICCS*, volume 954 of *LNCS*, pages 32–43. Springer, 1995.

[105] D. Leman, A. Feelders, and A. J. Knobbe. Exceptional model mining. In W. Daelemans, B. Goethals, and K. Morik, editors, *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II*, volume 5212 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2008.

[106] P. Lenca, P. Meyer, B. Vaillant, and S. Lallich. On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. *European Journal of Operational Research*, 184(2):610–626, 2008.

[107] C. C. Licon, G. Bosc, M. Sabri, M. Mantel, A. Fournel, C. Bushdid, J. Golebiowski, C. Robardet, M. Plantevit, M. Kaytoue, and M. Bensafi. Chemical features mining provides new descriptive structure-odor relationships. *PLOS Computational Biology*, 15(4):1–21, 04 2019.

[108] S. Lopes, J.-M. Petit, and L. Lakhal. Functional and approximate dependency mining: database and fca points of view. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2-3):93–114, 2002.

[109] C. Low-Kam, C. Raïssi, M. Kaytoue, and J. Pei. Mining statistically significant sequential patterns. In *IEEE 13th International Conference on Data Mining*, 2013.

[110] T. Lucas, T. C. P. B. Silva, R. Vimieiro, and T. B. Ludermir. A new evolutionary algorithm for mining top-k discriminative patterns in high dimensional data. *Appl. Soft Comput.*, 59:487–499, 2017.

[111] 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.

[112] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.

[113] M. Mampaey, S. Nijssen, A. Feelders, and A. J. Knobbe. Efficient algorithms for finding richer subgroup descriptions in numeric and nominal data. In *ICDM*, 2012.

[114] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[115] R. Mathonat, D. Nurbakova, J.-F. Boulicaut, and M. Kaytoue. SeqScout: Using a Bandit Model to Discover Interesting Subgroups in Labeled Sequences. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Washington, United States, 2019.

[116] M. Meeng, W. Duivesteijn, and A. J. Knobbe. Rocsearch - an roc-guided search strategy for subgroup discovery. In M. J. Zaki, Z. Obradovic, P. Tan, A. Banerjee, C. Kamath, and S. Parthasarathy, editors, *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 704–712. SIAM, 2014.

[117] B. Mirkin. *Mathematical classification and clustering*. Boston-Dordrecht: Kluwer academic publisher, 1996.

[118] B. Mirkin. *Clustering for data mining: a data recovery approach*. Chapman & Hall/Crc Computer Science, 2005.

[119] B. Mirkin and A. V. Kramarenko. Approximate bicluster and tricluster boxes in the analysis of binary data. In S. O. Kuznetsov, D. Slezak, D. H. Hepting, and B. Mirkin, editors, *Proceedings of the 13th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (RSFDGrC 2011)*, volume 6743 of *Lecture Notes in Computer Science*, pages 248–256. Springer, 2011.

[120] O. Missura and T. Gärtner. Predicting dynamic difficulty. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011*, pages 2007–2015, 2011.

[121] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[122] S. Moens and M. Boley. Instant exceptional model mining using weighted controlled pattern sampling. In *Proceedings IDA 2014*, pages 203–214, 2014.

[123] F. Monrose and A. D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 2000.

[124] S. Morishita and J. Sese. Traversing itemset lattice with statistical metric pruning. In *ACM SIGMOD-SIGACT-SIGART*, pages 226–236, 2000.

[125] S. Motameny, B. Versmold, and R. Schmutzler. Formal concept analysis for the identification of combinatorial biomarkers in breast cancer. In R. Medina and S. A. Obiedkov, editors, *Formal Concept Analysis, 6th International Conference (ICFCA)*, volume 4933 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2008.

[126] P. K. Novak, N. Lavrac, D. Gamberger, and A. Krstacic. CSM-SD: methodology for contrast set mining through subgroup discovery. *Journal of Biomedical Informatics*, 42(1):113–122, 2009.

[127] P. K. Novak, N. Lavrac, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403, 2009.

[128] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. A survey of real-time strategy game AI research and competition in starcraft. *IEEE Trans. Comput. Intellig. and AI in Games*, 5(4):293–311, 2013.

[129] M. H. Overmars and J. van Leeuwen. Maintenance of Configurations in the Plane. *J. Comput. Syst. Sci.*, 23(2):166–204, 1981.

[130] V. Pachón, J. M. Vázquez, J. L. Domínguez, and M. J. M. López. Multi-objective evolutionary approach for subgroup discovery. In E. Corchado, M. Kurzynski, and M. Wozniak, editors, *Hybrid Artificial Intelligent Systems - 6th International Conference, HAIS 2011, Wroclaw, Poland, May 23-25, 2011, Proceedings, Part II*, volume 6679 of *Lecture Notes in Computer Science*, pages 271–278. Springer, 2011.

[131] Z. Pawlak. Rough sets. *International Journal of Parallel Programming*, 11(5):341–356, 1982.

[132] A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. *IEEE Security & Privacy*, 2(5):40–47, 2004.

[133] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, 2001.

[134] R. G. Pensa, C. Leschi, J. Besson, and J.-F. Boulicaut. Assessment of discretization techniques for relevant pattern discovery from gene expression data. In M. J. Zaki, S. Morishita, and I. Rigoutsos, editors, *Proceedings of the 4th ACM SIGKDD Workshop on Data Mining in Bioinformatics (BIOKDD 2004)*, pages 24–30, 2004.

[135] J. Poelmans, D. I. Ignatov, S. O. Kuznetsov, and G. Dedene. Formal concept analysis in knowledge processing: A survey on applications. *Expert Syst. Appl.*, 40(16):6538–6560, 2013.

[136] J. Poelmans, S. O. Kuznetsov, D. I. Ignatov, and G. Dedene. Formal concept analysis in knowledge processing: A survey on models and techniques. *Expert Syst. Appl.*, 40(16):6601–6623, 2013.

[137] 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.

[138] C. Raïssi, J. Pei, and T. Kister. Computing closed skycubes. *PVLDB*, 3(1):838–847, 2010.

[139] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. Osborne/McGraw-Hill, Berkeley, CA, USA, 2nd edition, 2000.

[140] D. Rodríguez, R. Ruiz, J. C. Riquelme, and J. S. Aguilar-Ruiz. Searching for rules to detect defective modules: A subgroup discovery approach. *Inf. Sci.*, 191:14–30, 2012.

[141] S. Roman. *Lattices and Ordered Sets*. Springer New York, 2008.

[142] S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

[143] Y. Sagiv, C. Delobel, D. S. P. Jr., and R. Fagin. An equivalence between relational database dependencies and a fragment of propositional logic. *Journal of the ACM*, 28(3):435–453, 1981.

[144] M. P. D. Schadd, M. H. M. Winands, H. J. van den Herik, G. Chaslot, and J. W. H. M. Uiterwijk. Single-player monte-carlo tree search. In H. J. van den Herik, X. Xu, Z. Ma, and M. H. M. Winands, editors, *Computers and Games, 6th International Conference, CG 2008, Proceedings*, volume 5131 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2008.

[145] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[146] D. A. Simovici, D. Cristofor, and L. Cristofor. Impurity measures in databases. *Acta Inf.*, 38(5):307–324, 2002.

[147] D. A. Simovici and R. L. Tenney. *Relational Database Systems*. Academic Press, Inc., Orlando, FL, USA, 1st edition, 1995.

[148] A. Soulet, C. Raïssi, M. Plantevit, and B. Crémilleux. Mining dominant patterns in the sky. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 655–664, 2011.

[149] M. Stanescu and M. Certicky. Predicting opponent's production in real-time strategy games with answer set programming. *Computational Intelligence and AI in Games, IEEE Transactions on*, PP(99):1–1, 2014.

[150] P. Symeonidis, D. Ntempos, and Y. Manolopoulos. Location-based social networks. In *Recommender Systems for Location-based Social Networks*, SpringerBriefs in Electrical and Computer Engineering, pages 35–48. Springer New York, 2014.

[151] G. Synnaeve and P. Bessière. A bayesian model for opening prediction in RTS games with application to starcraft. In *2011 IEEE Conference on Computational Intelligence and Games, CIG 2011*, pages 281–288, 2011.

[152] T. L. Taylor. *Raising the Stakes:E-Sports and the Professionalization of Computer Gaming*. MIT Press, 2012.

[153] A. B. Tchagang, S. Phan, F. Famili, H. Shearer, P. R. Fobert, Y. Huang, J. Zou, D. Huang, A. Cutler, Z. Liu, and Y. Pan. Mining biological information from 3d short time-series gene expression data: the optricluster algorithm. *BMC Bioinformatics*, 13:54, 2012.

[154] J. J. Thompson, M. R. Blair, L. Chen, and A. J. Henrey. Video game telemetry as a critical tool in the study of complex skill learning. *PLoS ONE*, 2013.

[155] J. Ullman. *Principles of Database Systems and Knowledge-Based Systems, volumes 1–2*. Computer Science Press, Rockville (MD), USA, 1989.

[156] 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.

[157] D. van der Merwe, S. A. Obiedkov, and D. G. Kourie. Addintent: A new incremental algorithm for constructing concept lattices. In P. W. Eklund, editor, *Concept Lattices, Second International Conference on Formal Concept Analysis, ICFCA 2004, Sydney, Australia, February 23-26, 2004, Proceedings*, volume 2961 of *Lecture Notes in Computer Science*, pages 372–385. Springer, 2004.

[158] M. van Leeuwen. Interactive data exploration using pattern mining. In A. Holzinger and I. Jurisica, editors, *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics - State-of-the-Art and Future Challenges*, volume 8401 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2014.

[159] M. van Leeuwen and A. J. Knobbe. Diverse subgroup set discovery. *Data Min. Knowl. Discov.*, 25(2):208–242, 2012.

[160] M. van Leeuwen and A. Ukkonen. Discovering skylines of subgroup sets. In H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezný, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD Proceedings, Part III*, volume 8190 of *Lecture Notes in Computer Science*, pages 272–287. Springer, 2013.

[161] A. Von Eschen. Machine learning and data mining in call of duty (invited industrial talk). In *Machine Learning and Knowledge Discovery in Databases - European Conference*, 2014.

[162] G. Voutsadakis. Polyadic concept analysis. *Order*, 19(3):295–304, 2002.

[163] B. G. Weber and M. Mateas. Case-based reasoning for build order in real-time strategy games. In *Proceedings of the Fifth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2009*, pages 106–111. The AAAI Press, 2009.

[164] B. G. Weber and M. Mateas. A data mining approach to strategy prediction. In *Symposium on Computational Intelligence and Games*, 2009.

[165] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered Sets*, pages 445–470. Reidel, 1982.

[166] R. Wille. Why can concept lattices support knowledge discovery in databases? *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2-3):81–92, 2002.

[167] S. Wrobel. An algorithm for multi-relational discovery of subgroups. In H. J. Komorowski and J. M. Zytkow, editors, *Principles of Data Mining and Knowledge Discovery, First European Symposium, PKDD '97, Trondheim, Norway, June 24-27, 1997, Proceedings*, volume 1263 of *Lecture Notes in Computer Science*, pages 78–87. Springer, 1997.

[168] C. Xia, R. Schwartz, K. E. Xie, A. Krebs, A. Langdon, J. Ting, and M. Naaman. Citybeat: real-time social media visualization of hyper-local city data. In *WWW 2014, Companion Volume*, pages 167–170. ACM, 2014.

[169] R. V. Yampolskiy and V. Govindaraju. Behavioural biometrics: a survey and classification. *International Journal of Biometrics*, 2008.

[170] E. Q. Yan, J. Huang, and G. K. Cheung. Masters of control: Behavioral patterns of simultaneous unit group manipulation in starcraft 2. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI)*, 2015.

[171] Y. Yang, G. I. Webb, and X. Wu. Discretization methods. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 101–116. Springer, 2010.

[172] B. Zalik. An efficient sweep-line Delaunay triangulation algorithm. *Computer-Aided Design*, 37(10):1027–1038, 2005.

[173] L. Zhao and M. J. Zaki. Tricluster: an effective algorithm for mining coherent clusters in 3d microarray data. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 694–705, New York, NY, USA, 2005. ACM.

[174] S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

[175] A. Zimmermann and S. Nijssen. Supervised Pattern Mining and Applications to Classification. In *Frequent Pattern Mining*, pages 425–442. Springer, 2014.

Abstract

The process of collecting and analyzing data to answer predictive, explanatory, and decision-making issues has come to be known as "data science" for more than thirty years. Firstly used only by scientists, mainly by statisticians, the term is now widely used in the academics and industrial world. This can be explained in two ways: (i) data is ubiquitous, large, and varied, and (ii) there has been an awareness of the omniscient potential of data. The latter can be economic, societal, scientific, or related to health-care, and is based not only on the data that an entity has, but also on data that it can get (sensors, social networks, open data, etc., freely or not) making the data a black oil that still needs algorithms, methods and methodologies, to be properly refined.

One component of data science, Knowledge Discovery in databases (KDD), deals in particular with the Data-Information-Knowledge process with the aim of explaining relationships or discovering hidden properties. Opposed to a purely statistical approach, a family of methods has met an important success over the last twenty years: data-mining and especially pattern-mining. Their goal is to describe, summarize, raise hypotheses from data. In particular, pattern mining makes it possible to efficiently find regularities of various types (such as frequent patterns in a set of transactions, molecular sub-graphs characteristic of toxicity, locally co-expressed gene groups, etc.). In fact, where conventional approaches aim to validate or invalidate an hypothesis given a priori, the search of patterns is seen as an enumeration technique of all the possible hypotheses (a set of exponential size w.r.t the input data) verifying some given constraints or maximizing a certain interest for the expert. Once discovered, the best hypotheses can then be tested, validated or invalidated and ultimately validated as knowledge unit.

My scientific adventure began with the study of a binary relationship, very often illustrated by supermaket transaction data, linking customers and products they buy. How to make this relationship speak? What knowledge, behavioral habits, recommendations, etc. can we characterize?

This initial question allowed me to travel through different application fields (biology, neuroscience, social networks and video games analytics), seeking to implement or adapt data mining methods to try to understand some phenomena while properly formalizing data and patterns in the most rigorous way. This is the story of this manuscript, according to three main research axes: the formalism framing the methods (*Formal Concept Analysis*), the methodological and algorithmic aspects related in *Data mining*, and finally the *Knowledge Discovery* "in practice" through several concrete applications encountered during collaborations with other scientists or industrial partners.

Keywords: Knowledge Discovery in Databases, Pattern Mining, Formal Concept Analysis, Applications.

Résumé

Le processus qui permet de collecter des volumes de données puis de les analyser pour répondre à des questions à buts prédictifs, explicatifs et décisionnels, est apparu sous le vocable "science des données" (data science) il y a déjà plus de trente années. Accaparé d'abord par les scientifiques (notamment les statisticiens et largement pratiqué par les physiciens), ce terme connaît aujourd'hui un usage répandu dans le monde industriel et les collectivités. Cela s'explique de deux manières : (i) les données sont aujourd'hui omniprésentes, en grandes quantités, et variées, et (ii) il y a eu une prise de conscience du potentiel omniscient de ces données. Ce dernier peut être économique, sociétal, sanitaire ou encore scientifique, et se base non plus seulement sur des données qu'une entité possède, mais également sur des données qu'elle peut se procurer (capteurs, réseaux sociaux, données ouvertes open data, etc., gratuitement ou non) faisant de la donnée un or noir toujours trop peu raffiné.

Une composante de la science de données, la "découverte de connaissances" (DC ou Knowledge discovery in databases, KDD), traite en particulier de la chaîne Données–Informations–Connaissances avec le souci d'expliciter des relations ou propriétés enfouies. Se différenciant d'une approche purement statistique une famille de méthodes a connu un succès vaste ces vingt dernières années : la fouille de données sous-contraintes. Elles visent à décrire, résumer, soulever des hypothèses à partir de données. Notamment, la fouille de motifs permet de trouver de manière efficace des régularités de divers types (comme des motifs fréquents dans un ensemble de transactions, des sous-graphes moléculaires caractéristiques d'une toxicité, des groupes gènes localement co-exprimés, etc.). En fait, là où les approches classiques visent à valider ou invalider une hypothèse donnée a priori, la fouille de motifs se voit au contraire comme une technique d'énumération de toutes les hypothèses possibles vérifiant certaines contraintes ou encore maximisant un certain intérêt pour l'expert parmi un ensemble de taille exponentiel. Une fois découvertes, les meilleures hypothèses peuvent être alors testées, validées ou invalidées. On fait donc véritablement face à un processus de découverte d'hypothèses ayant le plus de chances d'être validées ensuite comme connaissances.

Mon initiation scientifique a commencé par l'étude d'une relation binaire, très souvent illustrée par le panier de la ménagère, liant clients et produits qu'ils achètent. Comment faire parler cette relation données ? Quelles connaissances, habitudes comportementales, recommandations, etc. peut-on extraire ?

Cette question initiale m'a alors permis de voyager à travers différents domaines applicatifs (biologie, neurosciences, réseaux sociaux et jeux-vidéo), cherchant à mettre en application ou adaptant des méthodes de fouille de données pour tenter comprendre des phénomènes tout en formalisant le plus rigoureusement possible le cadre dans lequel ces méthodes s'inscrivent. C'est donc cette histoire que je vais raconter dans ce manuscrit, selon trois axes principaux : le formalisme cadrant les méthodes avec l'*Analyse de Concepts Formels*, l'aspect méthodologique et algorithmique à travers la *Fouille de données*, et enfin la *Découverte de Connaissances* à travers plusieurs applications concrètes rencontrées lors de collaborations avec d'autres scientifiques ou industriels

Mots-clés: Découverte de connaissances, fouille de motifs, analyse de concepts formels, applications