AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Extraction de motifs séquentiels dans des données séquentielles multidimensionnelles et hétérogènes–Une application à l'analyse de trajectoires de patients

THÈSE

présentée et soutenue publiquement le 02 Juillet 2014

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Elias Egho

Composition du jury

Président :	Dominique LAURENT	Professeur, Université de Cergy Pontoise
Rapporteurs :	Céline ROBARDET	Maître de conferences, Université de Lyon
	Fosca GIANNOTTI	Professeur, Université de Pisa
Examinateurs :	Toon CALDERS	Professeur, Université Libre de Bruxelles
	Claire GARDENT	Directeur de recherche CNRS, LORIA
	Vincent LEMAIRE	Chercheur, Orange Labs
Directeurs :	Chedy RAISSI	Chargé de recherche, INRIA Nancy
	Nicolas JAY	Maître de conferences, Université de Lorraine
	Amedeo NAPOLI	Directeur de recherche CNRS, LORIA

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

Mis en page avec la classe thesul.

Remerciements

The completion of my doctoral thesis has been an amazing journey which would not have been possible without the support and encouragement of other people in Orpailleur and abroad.

This is to acknowledge everyone who has assisted me during my doctoral studies. The first and foremost acknowledgement is for my supervisor, Mr. Amedeo NAPOLI, for thinking of me capable enough to hire as a doctoral student and then agreeing on supervising me. I would like to thank him for being patient during my PhD and giving me useful feedback for all my work and ideas. I am very grateful to him for all of his help. I have also benefited greatly from other mentors Mr. Chedy RAISSI and Mr. Nicolas JAY for helping me in my work and providing me insightful career advice. I want to thank them for the constructive feedback that they have provided to this project over the years. They have both been wonderful collaborators. In various ways, they made my time in the field productive and successful. I have learned a lot about how to do interesting and rigorous research from them.

Moreover, I would like to thank the reviewers of my PhD thesis, Mme. Céline ROBARDET and Mme. Fosca GIANNOTTI for their detailed report and their insightful comments. I am also thankful to the other members of my PhD committee, Mr. Toon CALDERS, Mr. Dominique LAURENT, Mme. Claire GARDENT and Mr. Vincent LEMAIRE for their valuable suggestions while evaluating my research work.

During my PhD, I benefited greatly from having access to different environments and institutions. In different ways, each of these experiences taught me the process of research. How to think about problems, how to perform evaluations and how to assess the validity and consistency of arguments. They provided a rich and supportive environment which nurtured me. These institutions not only helped me academically but also gave me the international exposure to life style and helped me in personal growth. More specifically, I want to thank Mr. Toon Calders who has been my supervisor for the internship. It was a joy to be surrounded by people who worked on topics that were so much mainstream. I am grateful to him for being open and friendly to newcomers like me.

A very special thanks to my co-authors for working with me and exchanging different ideas for making my work being applicable to broader field of research and providing a different perspective to my work. They were knowledgeable, fascinating, engaged, and friendly— in short, they were tremendous colleagues in every sense.

Finally, I would like to thank my family who has been very supportive during these years with love and encouragement. My parents have not only been a source of moral support, they have also taught me skills that have made me a better researcher. I have learned so much from my parents and I appreciate all of the sacrifices that they have made for me. I learned from my parents to work hard, to stand up for myself, and to pick apart an argument. I am grateful to them both for being wonderful role models to me.

À mes parents .

Contents

Introduction **ix**

1 Knowledge Discovery in Databases ix
2 Healthcare Trajectory x
3 Contributions xi
4 Thesis Organization xiii

Chapter 1
Sequential Pattern Mining **1**

1.1 Itemset Mining 1
1.2 Association Rule Mining 5
1.3 Sequential Pattern Mining 6
1.4 Sequential Pattern Mining Methods 8
1.4.1 Horizontal sequence mining 9
1.4.2 Vertical Sequence Mining 12
1.4.3 Projection-Based Sequence Mining 16
1.4.4 Extension 19
1.5 Multi Dimensional and Multilevel Sequential Patterns Algorithms 24
1.5.1 Multi Dimensional Sequential Patterns Methods 24
1.5.2 Multi Level Sequential Patterns Methods 28
1.5.3 Multi Dimensional and Multi Level Sequential Patterns Methods 30
1.6 Discussion 31

Chapter 2
Mining Heterogeneous Multidimensional Sequential Patterns

2.1 Introduction 35
2.2 Problem Statement 36
2.2.1 An introductory example 36
2.2.2 Basic definitions 37
2.2.3 Most specific multidimensional sequential patterns 39

2.3 MMISP algorithm . . . 40
2.3.1 Step 1: Extracting all frequent elementary vectors: . . . 40
2.3.2 Step 2: Transformation of mds-database: . . . 45
2.3.3 Step 3: mds-patterns mining: . . . 47
2.4 Implementation and Experimental Validation . . . 47
2.4.1 Experiments on Synthetic Datasets . . . 48
2.5 Conclusion and perspective . . . 51

Chapter 3
On Projections of Sequential Pattern Structures

3.1 Introduction . . . 53
3.2 Formal Concept Analysis . . . 55
3.2.1 Formal Context . . . 55
3.2.2 Formal Concept . . . 55
3.2.3 Concept Lattice . . . 57
3.2.4 Algorithms . . . 59
3.3 Reducing the Representation Complexity of Concept Lattice . . . 59
3.3.1 Iceberg Concept Lattice . . . 59
3.3.2 Stability . . . 60
3.4 Summarizing Closed Sequential Patterns with Partial Order . . . 62
3.5 Classification and selection of interesting sequential patterns by using FCA . . . 64
3.6 Multidimensional Sequential Pattern Structures . . . 67
3.6.1 Pattern Structures . . . 67
3.6.2 Sequential Data . . . 67
3.6.3 Multidimensional Sequential Meet-semilattice . . . 68
3.7 Projections of Sequential Pattern Structures . . . 70
3.8 Conclusion . . . 71

Chapter 4
On Measuring Similarity for Sequences of Itemsets

4.1 Introduction . . . 73
4.2 Related Work . . . 75
4.2.1 Edit distance . . . 75
4.2.2 Dynamic Time Warping (DTW) . . . 76
4.2.3 Similarity Measure for Sequential Patterns (S^2MP) . . . 77
4.2.4 Longest Common Subsequence (LCS) . . . 77
4.2.5 All Common Subsequences (ACS) . . . 78

4.3 Preliminaries . . . 78
4.4 Longest And All Common Subsequences: A Comparison . . . 81
4.5 Counting All Distinct Subsequences . . . 81
4.6 Counting All Common Subsequences . . . 84
4.6.1 Dynamic Programming . . . 87
4.7 Complexity and Linial-Nisan Approximation Results . . . 88
4.7.1 Complexity . . . 88
4.7.2 Linial-Nisan Approximation . . . 90
4.8 Implementation and Experimental Validation . . . 92
4.8.1 Clustering Handwritten Assamese Symbols . . . 93
4.8.2 Linial-Nisan approximation . . . 96
4.8.3 Studying Scalability on Synthetic Data Sets . . . 97
4.9 Conclusion and Perspective . . . 98

Chapter 5
Detection and Classification of Healthcare Trajectory

5.1 Introduction . . . 101
5.2 PMSI French healthcare system . . . 102
5.3 Mining Healthcare Trajectories by using *MMISP* . . . 105
5.3.1 Mining healthcare trajectories . . . 105
5.3.2 *MMISP* versus Standard sequential pattern mining method . . . 107
5.3.3 $MMISP$ versus M^3SP . . . 108
5.4 Mining Healthcare Trajectory by using Sequential Pattern Structure . . . 109
5.5 Classification of Healthcare Trajectory by using FCA . . . 112
5.6 Clustering Healthcare Trajectories by using sim_{ACS} . . . 114
5.6.1 Clustering Healthcare Trajectories . . . 114
5.6.2 A Comparison Between sim_{ACS} and sim_{LCS} . . . 115
5.6.3 Linial-Nisan Approximation . . . 117
5.7 Conclusion . . . 118

Chapter 6
Conclusion and Perspectives

6.1 Summary of Contributions . . . 121
6.1.1 Extraction of Multidimensional and Heterogeneous Sequential Patterns . . . 122
6.1.2 Sequential Pattern Structure . . . 122
6.1.3 Classification of Multidimensional and Heterogeneous Sequential Patterns: 123
6.1.4 Clustering Of Sequences: . . . 123

6.2 Perspective 124
6.3 List of publications 125

Introduction

1 Knowledge Discovery in Databases

Over the past few years, there has been a huge outburst of data in all domains and many formats. Some of the data is available as a raw format shared only between certain communities where as some data are available on-line for sharing between larger communities. Nowadays, huge amount of data is being produced related to healthcare system. This system is a rich source of data as they produce massive amounts of data such as electronic medical records, clinical trial data, hospital records, administrative data etc. On the other hand, analysis of such data can help in further discovery of hidden knowledge. As a result, many analysts struggle to develop powerful means for analysis and interpretation of such data and for the extraction of interesting knowledge that could help in decision-making for improving the healthcare system.

Data Mining, also known as Knowledge Discovery in Databases (KDD), refers to the non-trivial extraction of implicit, previously unknown and potentially useful information from data in databases as described in Fayyad et al. [1996a]. Figure 1 illustrates the KDD process. It is divided into three major steps. The first step is called pre-processing, which prepares target data to apply the techniques of data mining. This step divides into three sub-steps which are selection, preprocessing and transformation of data. The main step of KDD process is data mining where different algorithms can be applied to discover hidden knowledge. After that comes another process called post-processing, which evaluates the mining result according to users' requirements and domain knowledge.

More precisely, the KDD can be divided in five steps Fayyad et al. [1996a], Fayyad et al. [1996b] and Dunham [2006]:

Selection: The data needed for the data mining process may be obtained from many different and heterogeneous data source. In this step, the data is collected from various sources such as database, files, non electronic sources (reports, books, etc.).

Preprocessing: This step consists of cleaning the collected data by removing the inconsistencies and duplications noise. Then, the data is combined in a common source like database, data warehouse or other repositories.

Transformation: The preprocessed data are transformed into processable format because the data mining algorithms work on certain input format.

Data mining: The data mining step deals with development and application of several algorithms to extract the information and patterns derived by the KDD process.

Interpretation/Evaluation: The information discovered with data mining need to be validated by a domain experts and analysts. In this step, the extracted information is repre-

sented to the analysts through visualization techniques to help them in understanding and interpreting the results.

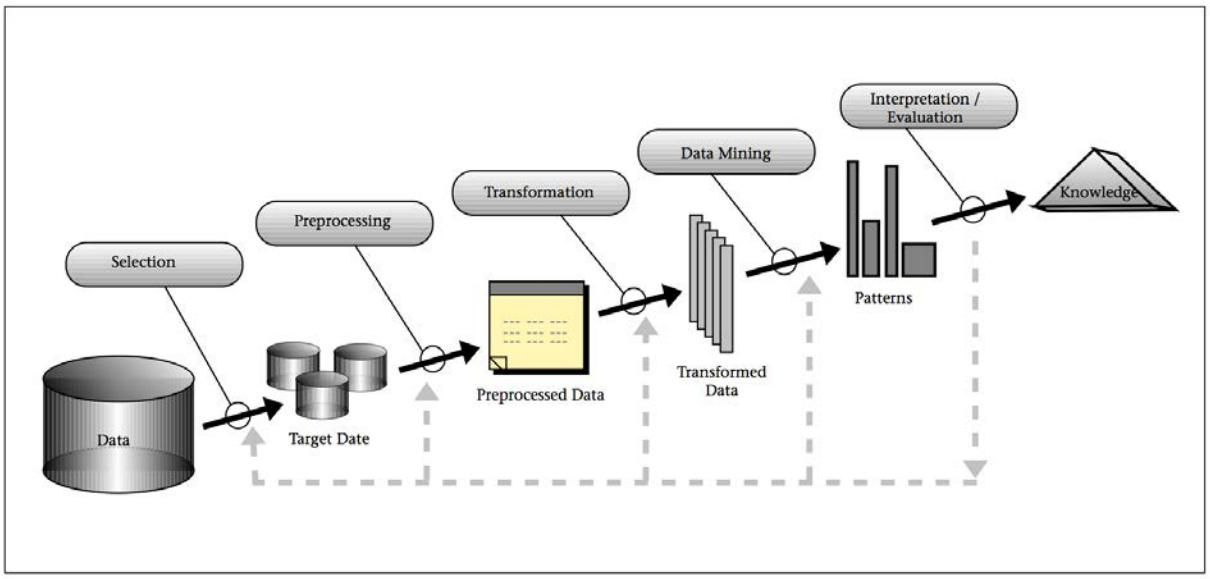


Figure 1: An Overview of the Steps That Compose the KDD Process.

However, the process of KDD is not a monolithic and univocal process in which it would apply a general principle to all types of stored or retrieved data. Depending on the types of information to be found, data mining step can be derived in several forms such as segmentation Jain and Dubes [1988], classification Weiss and Kulikowski [1991], the extraction of itemset and association rules Agrawal et al. [1993] and Agrawal and Srikant [1994], the extraction of more complex structures such as episodes, graphs Mannila et al. [1997] and Washio and Motoda [2003] or sequential pattern mining Agrawal and Srikant [1995] and Srikant and Agrawal [1996] which is the focus of current work.

2 Healthcare Trajectory

Health care systems of both developed and developing countries face challenges related to increase in life expectancy. Over the past 50 years, care of chronic diseases is considered as an important problem for the aging population. The prognosis and quality of life of patients with chronic conditions have evolved with the progress of medicine. In France, around 8 million of people are supported by health insurances undergoing long-term treatments. It implies that their situations require prolonged treatment and particularly costly care, e.g., diabetes or cancer diseases. These conditions are a real challenge for public health in the domain of medicine, economic, society as they are dangerous diseases. From an institutional point of view, chronic diseases often require a multidisciplinary approach. For example, diabetic patients during their treatment will regularly visit not only their general practitioner, but also an endocrinologist, a biologist, an ophthalmologist, and for some cases a cardiologist and/or a neurologist. Thus, chronic patients during their treatment are usually in contact with multiple stakeholders of the healthcare system. They receive a series of diagnostic tests, medical or surgical treatments. While, their health state follows a variable course to study their stability, decline, improvement, crisis, and undercurrent diseases.

These steps of treatment are formed a real *healthcare trajectory*. Controlling patient healthcare trajectory is important for several reasons as it guarantees the quality of care, life of patients, and medico-economic efficiency for the health care system. This task is based on the appropriate delivery of therapy, patient compliance, optimal monitoring, non-redundancy examinations and prevention of complications, which are ensured by strongly coordinated actors and care planning. The multidisciplinary approach leads professionals and health care organizations to work in a big network Jay [2008] and Jay et al. [2013].

In the healthcare system the first objective is to define a predetermined clinical pathway supported by patient during his treatment. To achieve this goal, group of experts develop guidelines for good clinical practice, built based on the recommendations from medical research. These experts standardize care of the patient from medical and scientifically proven facts. These guidelines can be used in a healthcare program in order to better orient the healthcare trajectories of patients. However, it is not clear that these recommendations are followed in practice. Furthermore, in a multidisciplinary approach, organizational factors can cause a malfunction in the implementation of programs of care.

In order to improve healthcare, it is necessary to observe, understand and model healthcare trajectories in reality. To effectively implement this process, information on the course of patients in a healthcare system is required. More precisely, this information is distributed among multiple sources such as manual or digital records kept by generalists or specialist doctor, departmental records, records stored in hospital information system. The computerization of medical records, interoperability of information systems, the deployment of personal health record should gradually resolve this process. However, if data are available, very few methods are available to describe and classify a set of trajectories of care. The main objective of our work is to describe healthcare trajectory for its better understanding. To achieve this goal, data mining methods are used to analyze the healthcare trajectories of set of patients with healthcare data.

3 Contributions

This thesis contributes in designing a system for knowledge discovery based on the analysis of multidimensional sequential data. This system helps in the detection and classification of patient healthcare trajectories with an application in the oncology domain. It analyses data from a medico-administrative database which consists of information about the hospitalizations of patients in the region of Lorraine, France. The system allows the identification of characteristics of the episodes of care. For example, sequences representing hospitalizations related to the treatment of certain forms of cancer. These sequences consist of surgery followed by chemotherapy and monitoring. Therefore, it is necessary to develop specific methods for mining and classification of episode of care.

Mining patients trajectories: Data mining methods are especially adapted to the analysis of sequences and were successfully used in biomedical domain Huang et al. [2012], Kim et al. [2011], Petitjean et al. [2011], Batal et al. [2009] and Jay et al. [2006]. Case mix[1] systems capture medical problems, procedures, demographic and administrative data using controlled vocabularies and standardized records. In that context, sequences of hospitalizations can be analyzed with sequential pattern mining algorithms Srikant and Agrawal

[1]The term *case mix* refers to the type or mix of patients treated by a hospital or unit.

[1996]. Meanwhile, case mix records have a multidimensional structure that traditional sequential patterns can not fully reflect. Moreover, the granularity of the initial data may be too fine to generate interesting patterns. The availability of classifications used to code information in case mix systems is an opportunity to integrate additional knowledge into the mining process and achieve better results. Although a few approaches have been developed to tackle the problems of granularity and multidimensionality in sequential pattern mining Plantevit et al. [2010], they are still not adapted to the problem of mining care trajectories. For this reason, we propose a new approach to mine healthcare trajectories. We provide formal definitions and propose a new algorithm *MMISP* to mine this kind of sequences. The *MMISP* algorithm relies on external taxonomies to improve the mining process and produces results with appropriate levels of granularity. We conduct experiments on both real-world and synthetic datasets. The method is applied on real-world data where the problem is to mine healthcare patient trajectories and give potential interesting patterns for healthcare specialists and analysts.

Classification of patient trajectory: Formal Concept Analysis (FCA) is a method of data analysis and knowledge representation which processes information in the form of concept hierarchies constructed from binary tables Wille [1982]. FCA is a method of unsupervised classification. It is able to discover and organize the natural groupings of a set of objects linked to a set of attributes. FCA produces a concept lattice which facilitates the task of the analyst in knowledge discovery. We use FCA as a lattice-based classification method to regroup the healthcare trajectories mined with *MMISP*. This experiment provides two advantages. First, it helps in grouping patient trajectories sharing some attributes and produce their graphical visualizations. The second benefit of using FCA is that it is able to characterize groups of patients by several trajectories, and finally, it helps in pruning concept lattices with the help of support and stability to filter out only interesting group of patient trajectories.

Sequential Pattern Structures: Pattern structures consist of objects with their descriptions that allow a semilattice operation on them Ganter and Kuznetsov [2001b]. The problem with *MMSIP* method and all multidimensional sequential pattern mining algorithms is that it is impossible to find a sequential pattern that contains an event which is comparable with another event in the same pattern or in the another one. This limitation does not allow the discovery of patterns going from precise to general knowledge or vice versa. To echo this challenge, we propose a novel way to mine multidimensional sequences by mapping them to pattern structures. The genericity power provided by the pattern structures allows our approach to be directly instantiated with state-of-the-art FCA algorithms, making the final implementation flexible, accurate and scalable. Pattern structures can be hard to process usually due to the large number of concepts in the concept lattice and the complexity of the involved descriptions and the similarity operation. To solve this problem, we introduce and discuss the notion of *projections* for sequential pattern structures. These mathematical objects significantly decrease (i.e., filter) the number of patterns, while preserving the most interesting ones for an expert. Moreover, the projection provides an efficient tool for the analysis of multidimensional sequential datasets. Sequential pattern structure is applied as a mining method on real-world data where the problem is to mine healthcare patient trajectories and extract some interesting frequent healthcare trajectories to answer the questions of healthcare managers and decision makers.

Clustering patient trajectory: Clustering is the process of grouping a set of physical or ab-

stract objects into classes of similar objects Han et al. [2006]. In clustering, objects are grouped together based on their similarities. However, for a large part of the literature, similarity measures on sequential data remain limited to simple sequences, which are ordered lists of items (i.e., symbols) Levenshtein [1966], Herranz et al. [2011], Keogh [2002], Wang and Lin [2007]. By contrast, in healthcare system, patient trajectories are represented as ordered lists of itemsets (i.e., sets of symbols). This peculiarity is in itself a challenge as it implies to carefully take into account complex combinatorial aspects to compute similarities between sequences. To solve this problem, we focus on the notion of common subsequences as a means to define a distance or similarity score between a pair of sequences composed of a list of itemsets. The similarity between two sequences is defined as the ratio between the number of common subsequences from two sequences S and T divided by the maximal number of distinct subsequences. We discuss and present a dynamic programming algorithm for counting all common subsequences between two given sequences and all distinct subsequences for a given sequence. We notice that our silmilarity measure relies heavily on the inclusion-exclusion principle. The computation drawback is the fact that the inclusion-exclusion formula has an exponential number of terms which can become a problem with very long sequences. This prompted our interest in approximating our similarity measure through the approximation of the inclusion-exclusion formula used in both counting all common subsequences between two given sequences and counting all distinct subsequences for a given sequence. To solve this problem and to cope with large data sets containing long input sequences, an *approximation technique* is proposed to compute the similarity efficiently. The approach relies on approximating the size of a union of a family of sets in terms of the intersections of all subfamilies (i.e., *inclusion-exclusion principle*) based on the direct application of a result from Linial and Nisan [1990]. We report an extensive empirical study on synthetic datasets and qualitative experiments with datasets consisting of trajectories of cancer patients extracted from French healthcare organizations and online handwritten Assamese characters. We give empirical evidence showing that the proposed approximation method works well in practice. The visualizations of the results obtained by our similarity measure is available at this website `http://www.loria.fr/~eegho/acs/`.

4 Thesis Organization

This thesis is structured as follows:

Chapter 1 gives an overview of the studies related to the problem of mining sequential patterns.

Chapter 2 focuses on the problem of mining heterogeneous multidimensional sequential patterns. We generalize the concept of multidimensional sequence by considering heterogeneous multidimensional sequences. We propose a new method *MMISP (Mining Multidimensional Itemsets Sequential Patterns)* to extract sequential patterns from heterogeneous multidimensional sequential database. In addition, the approach is able to take into account background knowledge lying in taxonomies existing for each dimension. Several experiments over synthetic datasets are conducted to highlight the fact that *MMISP* is efficient in terms of runtime for a large panel of sequences with varying parameters.

Chapter 3 focuses on the theoretical foundations of Formal Concept Analysis. We show interesting properties of concept lattices and stability index to select interesting formal concepts. The classification of patient healthcare trajectories is achieved by taking advantage of the classifica-

tion capabilities of FCA. Then, two measures, support and stability, are used to filter out only interesting group of patient healthcare trajectories. We propose also a novel way of dealing with multidimensional sequences by mapping them to pattern structures. Then, we introduce and discuss the notion of *"projections"* for sequential pattern structures which significantly decreases (i.e., filter) the number of patterns, while preserving the most interesting ones for an expert.

Chapter 4 discusses the notion of common subsequences as a means to define a distance or similarity score between a pair of sequences composed of a list of itemsets. We introduce a measure which is the ratio between the number of common subsequences from any two sequences divided by the maximal number of distinct subsequences. A complete study of the complexity of the problem and an efficient approximation technique is presented. The quality and scalability of our measure is studied with the help of two experiments. The first one focus on clustering handwritten Assamese symbols from the UCI Machine Learning Repository. While, the second experiment is conducted over several synthetic datasets with different parameters to present the computation time and the scalability of our measure.

Chapter 5 firstly presents PMSI system, the French case mix information system. In order to improve patients healthcare trajectories in this system, we apply our approaches on these trajectories. Thus, this chapter focuses on results obtained by applying our approaches to understand, classify or cluster patients healthcare trajectories.

This document is concluded with a summary of contributions and some perspectives of our work.

Chapter 1

Sequential Pattern Mining

Contents

1.1 Itemset Mining 1
1.2 Association Rule Mining 5
1.3 Sequential Pattern Mining 6
1.4 Sequential Pattern Mining Methods 8
1.4.1 Horizontal sequence mining 9
1.4.2 Vertical Sequence Mining 12
1.4.3 Projection-Based Sequence Mining 16
1.4.4 Extension 19
1.5 Multi Dimensional and Multilevel Sequential Patterns Algorithms 24
1.5.1 Multi Dimensional Sequential Patterns Methods 24
1.5.2 Multi Level Sequential Patterns Methods 28
1.5.3 Multi Dimensional and Multi Level Sequential Patterns Methods 30
1.6 Discussion 31

Pattern mining is one of the important fields in data mining. A pattern can be considered as a set of events that occur more than once in a particular data set. This chapter aims at summarizing all studies that have been conducted to date for mining sequential pattern. Before discussing the new methods proposed in this thesis, we introduce some basic definitions and topics which are currently and frequently researched in the field of pattern mining.

1.1 Itemset Mining

Frequent itemset mining is a data analysis method that was originally developed for market basket analysis. It aims at finding regularities in the shopping behavior of the customers of supermarkets, mail order companies, and online shops. In particular, it tries to identify sets of products that are frequently bought together. Once identified, such sets of associated products may be used to optimize organization of the offered products on the shelves of a supermarket, the pages of a mail order catalogue, web shop or provide suggestions about which products may conveniently be bundled together. Given a set $\mathcal{I} = \{i_1, ..., i_n\}$ of items, called the item base and a database of transactions. An item may, for example, represent a product. In this case, the item base represents the set of all products offered by a supermarket. The term itemset refers

to any subset of $\mathcal{I}$. Each transaction in the database is an itemset and may represent, in the supermarket setting, a set of products that has been bought by a customer. As several customers may have bought the same set of products. More formally,

Definition 1. *(Itemset)*
Let $\mathcal{I} = \{i_1, ..., i_n\}$ be a finite set of items. *An* itemset *$x = \{x_1, ..., x_k\}$ is a non-empty subset of $\mathcal{I}$. k is called the size of the itemset x, denoted by $|x| = k$.*

Definition 2. *(Transactional Database)*
Transactional database $\mathcal{T}_{\mathcal{DB}} = \{(id_1, t_{id_1}), (id_2, t_{id_2})..., (id_m, t_{id_m})\}$ over $\mathcal{I} = \{i_1, ..., i_n\}$ is defined as a set of m pairs (id, t_{id}) where id is an identification ($id \in \mathbb{N}$) and t_{id} is an itemset $t_{id} \subseteq \mathcal{I}$.

The support of an itemset is usually defined as the number of transactions in the dataset in which a particular itemset occurs. A formal definition of the support of itemset is as follows:

Definition 3. *(Support of Itemset)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database and let $x = \{x_1, x_2, ..., x_k\}$ be an itemset. The support of x, denoted by $support(x, \mathcal{T}_{\mathcal{DB}})$ is defined as follows:

$$support(x, \mathcal{T}_{\mathcal{DB}}) = |\{(id, t_{id}) \in \mathcal{T}_{\mathcal{DB}};\ x \subseteq t_{id}\}|$$

The collection of frequent itemset in $\mathcal{T}_{\mathcal{DB}}$ with respect to a user support threshold σ_s is defined as follows:

Definition 4. *(Frequent Itemset)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database over a set of items $\mathcal{I}$, and σ_s a minimal support threshold. The itemset x ($x \subseteq \mathcal{I}$) is a frequent itemset if and only if $support(x, \mathcal{T}_{\mathcal{DB}}) \geqslant \sigma_s$

Definition 5. *(Set of Frequent Itemsets)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database over a set of items $\mathcal{I}$, and σ_s a minimal support threshold. The collection of frequent itemsets in $\mathcal{T}_{\mathcal{DB}}$ with respect to σ_s is denoted by:

$$\mathcal{FI}(\mathcal{T}_{\mathcal{DB}}, \sigma_s) = \{x \subseteq \mathcal{I};\ support(x, \mathcal{T}_{\mathcal{DB}}) \geqslant \sigma_s\}$$

Frequent itemset mining can be defined as follows:

Definition 6. *(Frequent Itemset Mining)*
Given a set of items $\mathcal{I}$, a transactional database $\mathcal{T}_{\mathcal{DB}}$ over $\mathcal{I}$, and minimal support threshold σ_s, The frequent itemset mining is to find the set $\mathcal{FI}(\mathcal{T}_{\mathcal{DB}}, \sigma_s)$.

Example 1. *Considering the transactional database $\mathcal{T}_{\mathcal{DB}}$ shown in Table 1.1.a over the set of items $\mathcal{I} = \{beer, coke, bread, chips, butter\}$. Table 1.1.b shows all frequent sets in $\mathcal{T}_{\mathcal{DB}}$ with respect to a minimal support threshold equal to 2.*

The search space about frequent itemsets is naturally made up of the subset relationships between itemsets, which form a *partial order* on $2^{|\mathcal{I}|}$ Agrawal and Srikant [1994]. The partial order can be seen as a *Hasse diagram* Burdakov et al. [2005], which is essentially a graph, in which each itemset $x \subseteq \mathcal{I}$ form a node and the edge between two nodes of itemsets x,y represents a "$\subset$" relation. Hence, the edge represents $x \subset y$ and $\nexists z;\ x \subset z \subset y$.

$\mathcal{T}_{\mathcal{DB}} =$

1	$\{beer, chips\}$
2	$\{beer, coke, chips\}$
3	$\{bread, butter\}$
4	$\{beer, chips\}$
5	$\{beer, coke, bread\}$

a. A market basket dataset

$\mathcal{FI}(\mathcal{T}_{\mathcal{DB}}, 2) =$

Frequent Itemset	Support
$\{beer\}$	4
$\{chips\}$	3
$\{beer, chips\}$	3
$\{bread\}$	2
$\{beer\}$	2
$\{coke\}$	2
$\{beer, coke\}$	2

b. Frequent itemsets extracted with a user support threshold $\sigma_s = 2$

Table 1.1: An example of itemset mining.

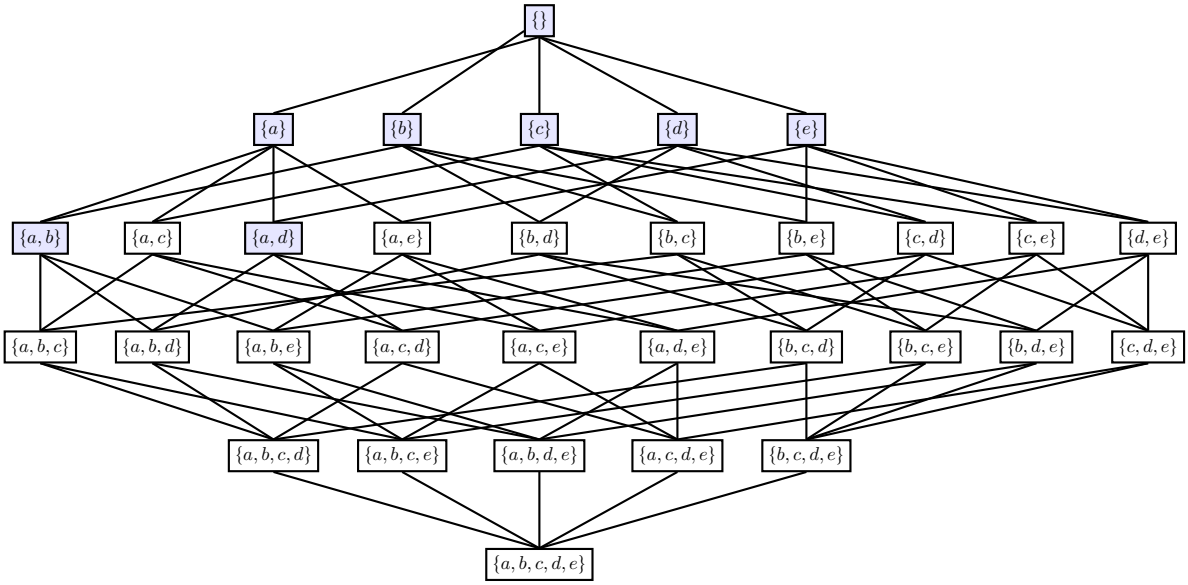


Figure 1.1: Hasse diagram for the partial order induced by $\subseteq$ on $2^{|\{a,b,c,d,e\}|}$ and $\sigma_s = 2$.

Example 2. *For the sake of simplicity in the representation, we assign each element of* $\mathcal{I} = \{beer, coke, bread, chips, butter\}$ *an alphabet, i.e,* $\mathcal{I} = \{a, b, c, d, e\}$. *Figure 1.1 shows a Hasse diagram based on partial order of* $2^{|\mathcal{I}|}$ *for* $\mathcal{I} = \{a, b, c, d, e\}$. *The blue nodes represent the frequent itemsets extracted from transactional database Table 1.1.a with a given support threshold* $\sigma_s = 2$ *(see Table 1.1.b).*

It is computationally not feasible to generate every candidate item set in the power set $2^{|\mathcal{I}|}$, determine its support and filter out infrequent sets as even small supermarkets usually keep thousands of different products. In the best case scenario, only the frequent sets are generated and counted. In order to make the search more efficient, one exploits a fairly obvious property of itemset support, namely that it is *antimonotone*.

Property 1. *(Antimonotone)*
Given a transactional database $\mathcal{T}_{\mathcal{DB}}$ *over* $\mathcal{I}$*,and two itemsets* x *and* y*. Then,*

$$\forall x, y \subseteq \mathcal{I};\ x \subseteq y \Rightarrow support(x, \mathcal{T}_{\mathcal{DB}}) \geqslant support(y, \mathcal{T}_{\mathcal{DB}}).$$

Hence, if a set is infrequent, all of its supersets must be infrequent and vice versa, else if a set is frequent, all of its subsets must be frequent too. This antimonotonicity property is also called as *downward closure* property, since the set of frequent sets is downward closed with respect to set inclusion. Similarly, the set of infrequent sets is upward closed. Together with the user support threshold, we immediately obtain the property *Apriori*.

Property 2. *(Apriori)*
Given a transactional database $\mathcal{T}_{\mathcal{DB}}$ over $\mathcal{I}$, two itemsets x and y, and minimal support threshold σ_s. Then,

$$(x \; is \; not \; frequent) \; \wedge \; (x \subseteq y) \Rightarrow (y \; is \; not \; frequent).$$

Because of the discovery of *Apriori* property, the research devoted to *frequent itemset mining* has great positive impact as it paved way for a variety of sophisticated and efficient algorithms like *Apriori Algorithm* Agrawal and Srikant [1994] Agrawal et al. [1996], *Eclat* Zaki et al. [1997] Schmidt-Thieme [2004], *FP-Growth* (Frequent Pattern Growth) Han et al. [2000b] Zhu and Grahne [2004], SaM (Split and Merge) Borgelt and Wang [2009] , RElim Borgelt [2010], H-Mine Pei et al. [2004a] .

When mining frequent itemsets the output is often huge and may even exceed the size of the transaction database to mine. In order to reduce this output without any loss of information, several approaches were designed. The most basic of these approaches is to restrict the output to so-called *closed frequent itemsets* Pasquier et al. [1999], Pan et al. [2003], *maximal frequent itemsets* Bayardo [1998], Burdick et al. [2001], *generator frequent itemsets* Pasquier et al. [1999], Bastide et al. [2000], Liu et al. [2008].

Definition 7. *(Closed Frequent Itemsets)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database over a set of items $\mathcal{I}$, and σ_s a minimal support threshold. The collection of closed frequent itemsets in $\mathcal{T}_{\mathcal{DB}}$ with respect to σ_s is denoted by:

$$\mathcal{CFI}(\mathcal{T}_{\mathcal{DB}}, \sigma_s) = \{x \subseteq \mathcal{I}; support(x, \mathcal{T}_{\mathcal{DB}}) \geqslant \sigma_s \wedge \; \forall y \supset x; \; support(y, \mathcal{T}_{\mathcal{DB}}) < support(x, \mathcal{T}_{\mathcal{DB}})\}$$

Definition 8. *(Maximal Frequent Itemsets)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database over a set of items $\mathcal{I}$, and σ_s a minimal support threshold. The collection of maximal frequent itemsets in $\mathcal{T}_{\mathcal{DB}}$ with respect to σ_s is denoted by:

$$\mathcal{MFI}(\mathcal{T}_{\mathcal{DB}}, \sigma_s) = \{x \subseteq \mathcal{I}; support(x, \mathcal{T}_{\mathcal{DB}}) \geqslant \sigma_s \wedge \; \forall y \supset x; \; support(y, \mathcal{T}_{\mathcal{DB}}) < \sigma_s\}$$

Definition 9. *(Frequent Itemsets Generator)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database over a set of items $\mathcal{I}$, and σ_s a minimal support threshold. The collection of generator frequent itemsets in $\mathcal{T}_{\mathcal{DB}}$ with respect to σ_s is denoted by:

$$\mathcal{GFI}(\mathcal{T}_{\mathcal{DB}}, \sigma_s) = \{x \subseteq \mathcal{I}; \nexists y \subseteq x \wedge \; support(x, \mathcal{T}_{\mathcal{DB}}) = support(y, \mathcal{T}_{\mathcal{DB}})\}$$

A-Close, an algorithm for mining frequent closed itemsets was proposed in Pasquier et al. [1999]. Other closed pattern mining algorithms include *CLOSET* Pei et al. [2000], *CHARM* Zaki and Hsiao [2002], *CLOSET+* Wang et al. [2003], *FPClose* Grahne and Zhu [2005] and *AFOPT* Liu et al. [2003]. Bayardo [1998] proposes an algorithm *MaxMiner* for mining maximal frequent itemsets from large databases. There are several other efficient methods for mining maximal frequent itemsets such as *IsTa* Borgelt et al. [2011], *Carpenter* Pan et al. [2003] and *GenMax* Bayardo [1998].

1.2 Association Rule Mining

Association rule mining is one of the most important and well researched topics in data mining for the extraction of interesting correlations, frequent patterns, associations or casual structures among frequent itemsets in the transaction databases Agrawal and Srikant [1994] and Agrawal et al. [1996]. However, using frequent itemsets is not always enough. For example, in the market basket analysis, besides extracting frequent itemsets it is also important to know which items help in determining this correlation. In Table 1.1.a, we can easily observe that each customer who bought *coke* also bought *beer*, but not vice versa. This means that we can predict with 100% confidence that a customer who buys *coke* will also buy beer, but the chance that a customer who buys *beer* will also buy *coke* is just 50%.

Formally, an association rule of the form $x \Rightarrow y$, where $x \subseteq \mathcal{I}$, $y \subseteq \mathcal{I}$ and $x \cap y = \varnothing$, tells us that an occurrence of itemset x implies the occurrence of itemset y in the same transaction. The rule $x \Rightarrow y$ holds in the transaction set $\mathcal{I}$ with *confidence* which is the number of transaction in $\mathcal{T}_{\mathcal{DB}}$ that contain x also contain y Agrawal and Srikant [1994], Agrawal et al. [1996]. The *support* of the association rule is the number of transactions in $\mathcal{T}_{\mathcal{DB}}$ containing $x \cup y$ Agrawal and Srikant [1994] and Agrawal et al. [1996]. The confidence and support of a rule is defined as follows.

Definition 10. *(Confidence of Rule)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database and let $x \Rightarrow y$ be a rule. The confidence of $x \Rightarrow y$, denoted by $confidence(x \Rightarrow y, \mathcal{T}_{\mathcal{DB}})$, is defined as follows:

$$confidence(x \Rightarrow y, \mathcal{T}_{\mathcal{DB}}) = \frac{support(x \cup y, \mathcal{T}_{\mathcal{DB}})}{support(x, \mathcal{T}_{\mathcal{DB}})}$$

Definition 11. *(Support of Rule)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database and let $x \Rightarrow y$ be a rule. The support of $x \Rightarrow y$, denoted by $support(x \Rightarrow y, \mathcal{T}_{\mathcal{DB}})$, is defined as follows:

$$support(x \Rightarrow y, \mathcal{T}_{\mathcal{DB}}) = \frac{support(x \cup y, \mathcal{T}_{\mathcal{DB}})}{|\mathcal{T}_{\mathcal{DB}}|}$$

The problem of association rule mining is defined as follows:

Definition 12. *(Set of Association Rules)*
Let $\mathcal{T}_{\mathcal{DB}}$ be a transactional database over a set of items $\mathcal{I}$, σ_s a minimal support threshold and σ_c minimal confidence threshold. The collection of association rules in $\mathcal{T}_{\mathcal{DB}}$ with respect to σ_s and σ_c is denoted by:

$$\mathcal{AR}(\mathcal{T}_{\mathcal{DB}}, \sigma_s, \sigma_c) = \{x \Rightarrow y \,; x, y \subseteq \mathcal{I} \wedge support(x \Rightarrow y, \mathcal{T}_{\mathcal{DB}}) \geqslant \sigma_s \wedge confidence(x \Rightarrow y, \mathcal{T}_{\mathcal{DB}}) \geqslant \sigma_c\}$$

Definition 13. *(Association Rules Mining)*
Given a set of items $\mathcal{I}$, a transactional database $\mathcal{T}_{\mathcal{DB}}$ over $\mathcal{I}$, minimal support threshold σ_s and minimal confidence threshold σ_c. The problem of mining association rules is to find the set $\mathcal{AR}(\mathcal{T}_{\mathcal{DB}}, \sigma_s, \sigma_c)$.

Example 3. *Let us consider a transactional database $\mathcal{T}_{\mathcal{DB}}$ shown in Table 1.1.a over the set of items $\mathcal{I} = \{beer, coke, bread, chips, butter\}$. Table 1.2 shows all association rules in $\mathcal{T}_{\mathcal{DB}}$ with respect to a minimal support threshold equals to 0.4 (i.e. $\sigma_s = \frac{2}{5}$) and a minimal confidence threshold equals to 0.6 (i.e. $\sigma_c = \frac{3}{5}$).*

$\mathcal{AR}(\mathcal{T}_{\mathcal{DB}}, \frac{2}{5}, \frac{3}{5}) =$

Rules	Support	Confidence
$\{beer\} \Rightarrow \{chips\}$	0.6	0.75
$\{chips\} \Rightarrow \{beer\}$	0.6	1
$\{coke\} \Rightarrow \{beer\}$	0.4	1

Table 1.2: Association rules extracted with a user support threshold $\sigma_s = 0.4$ and a user confidence threshold $\sigma_c = 0.6$ in the transactional database $\mathcal{T}_{\mathcal{DB}}$ shown Table 1.1.a.

$\mathcal{S}_{\mathcal{DB}} =$

1	$\langle\{a\}\{e\}\rangle$
2	$\langle\{a\}\{b,c,d\}\rangle$
3	$\langle\{a,c\}\rangle$
4	$\langle\{a\}\{b,c,d\}\{e\}\rangle$
5	$\langle\{e\}\rangle$

a. A Sequential database

$\mathcal{FSP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s) =$

Sequence	Support
$\langle\{a\}\rangle$	4
$\langle\{c\}\rangle$	3
$\langle\{e\}\rangle$	3
$\langle\{b\}\rangle$	2
$\langle\{d\}\rangle$	2
$\langle\{b,c\}\rangle$	2
$\langle\{b,c,d\}\rangle$	2
$\langle\{b,d\}\rangle$	2
$\langle\{c,d\}\rangle$	2

Sequence	Support
$\langle\{a\}\{b\}\rangle$	2
$\langle\{a\}\{c\}\rangle$	2
$\langle\{a\}\{d\}\rangle$	2
$\langle\{a\}\{e\}\rangle$	2
$\langle\{a\}\{b,c\}\rangle$	2
$\langle\{a\}\{b,d\}\rangle$	2
$\langle\{a\}\{c,d\}\rangle$	2
$\langle\{a\}\{b,c,d\}\rangle$	2

b. Frequent sequential patterns extracted with a user support threshold $\sigma_s = 2$

Table 1.3: An example of sequential pattern mining.

There have been several other research studies related to a variety of sophisticated and efficient algorithms for *association rules mining* , e.g., Webb [2000], Bayardo et al. [2000], Zaki [2000b], Aggarwal and Yu [2001], DuMouchel and Pregibon [2001], Omiecinski [2003], Ozgur et al. [2004], Wu et al. [2004], Kotsiantis and Kanellopoulos [2006], Ordonez et al. [2006], Lenca et al. [2008], Zhao et al. [2009], Herawan and Deris [2011], Nebot and Berlanga [2012], Sadh and Shukla [2013].

1.3 Sequential Pattern Mining

Itemsets and association rules are not the only patterns we can look for in a dataset. If the data is sequential by nature, itemsets do not possess the ability to express the order the events based on their occurrence, which is a major limitation. Assume that our database consists of sequences of itemsets, rather than transactions. An example of such a dataset is given in Table 1.3

If we apply traditional itemset mining to this example, given a support threshold of 2, we can see that $\{b, c, d\}$ and $\{a\}$ are frequent itemsets. However, all sequential information would be lost. If we look closely and take into account the sequential relation between itemsets, we can see that $\{b, c, d\}$ comes after $\{a\}$ frequently. Following are the basic definitions related to sequential pattern mining.

Definition 14. *(Sequence)*

Let $\mathcal{I} = \{i_1, ..., i_n\}$ be a finite set of items. *A* sequence *s over $\mathcal{I}$ is an ordered list $\langle s_1 \cdots s_l \rangle$, where s_i $(1 \leqslant i \leqslant l$, $l \in \mathbb{N})$ is an itemset. l is called the size of the sequence s, denoted by $|s| = l$. The length, denoted by $\ell(s)$, is the total number of items occurring in the sequence, i.e., $\ell(s) = \sum_{i=1}^{n} |s_i|$.*

Definition 15. *(Sequential Database)*
Sequential database $\mathcal{S}_{\mathcal{DB}} = \{(id_1, s_{id_1}), (id_2, s_{id_2}), ..., (id_m, s_{id_m})\}$ *is defined as a set of m pairs* (id, s_{id}) *where* s_{id} *is a sequence and id is a sequence identifier .*

Definition 16. *(Inclusion)*
A sequence $s = \langle s_1 \cdots s_l \rangle$ *is a subsequence of* $s' = \langle s'_1 \ldots s'_{l'} \rangle$*, denoted by* $s \subseteq s'$*, if there exist indices* $1 \leqslant i_1 < i_2 < \cdots < i_l \leqslant l'$ *such that* $s_j \subseteq s'_{i_j}$ *for all* $j = 1 \ldots l$ *and* $l \leqslant l'$*. s′ is said to be a supersequence of s.*

The support of a sequence is usually defined as the number of sequences in the sequential dataset which are a super sequence of the sequence . Support of sequence is defined as follows:

Definition 17. *(Support of Sequence)*
Let $\mathcal{S}_{\mathcal{DB}}$ *be a sequential database and let* $s = \langle s_1 \cdots s_l \rangle$ *be a sequence. The support of s, denoted by* $support(s, \mathcal{S}_{\mathcal{DB}})$ *is defined as follows:*

$$support(s, \mathcal{S}_{\mathcal{DB}}) = |\{(id, s_{id}) \in \mathcal{S}_{\mathcal{DB}};\ s \subseteq s_{id}\}|$$

A collection of sequential pattens in $\mathcal{S}_{\mathcal{DB}}$ with respect to a user defined support threshold σ_s is defined as follows:

Definition 18. *(Sequential Pattern)*
Given a set of items $\mathcal{I}$*, a sequential database* $\mathcal{S}_{\mathcal{DB}}$ *over* $\mathcal{I}$ *and minimal support threshold* σ_s*. The sequence s is called a sequential pattern if and only if* $support(s, \mathcal{S}_{\mathcal{DB}}) \geqslant \sigma_s$*.*

Definition 19. *(Set of Sequential Pattern)*
Let $\mathcal{S}_{\mathcal{DB}}$ *be a sequential database over a set of items* $\mathcal{I}$*, and* σ_s *a minimal support threshold. The collection of sequential patterns in* $\mathcal{S}_{\mathcal{DB}}$ *with respect to* σ_s *is denoted by:*

$$\mathcal{SP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s) = \{s;\ support(s, \mathcal{S}_{\mathcal{DB}}) \geqslant \sigma_s\}$$

Sequential pattern mining can be defined as:

Definition 20. *(Sequential Pattern Mining)*
Given a set of items $\mathcal{I}$*, a sequential database* $\mathcal{S}_{\mathcal{DB}}$ *over* $\mathcal{I}$*, and minimal support threshold* σ_s*, The sequential pattern mining is defined to find the set* $\mathcal{SP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s)$*.*

Example 4. *Consider a sequential database* $\mathcal{S}_{\mathcal{DB}}$ *shown in Table 1.3.a over the set of items* $\mathcal{I} = \{a, b, c, d, e\}$*. Table 1.3.b shows all frequent sequential patterns in* $\mathcal{S}_{\mathcal{DB}}$ *with respect to a minimal support threshold equal to 2.*

A naive approach for extracting sequential patterns is to generate all possible sequential patterns, compute their support and filter infrequent sequential patterns. Unfortunately, this approach can not be processed in a reasonable time. In order to make this search efficient, *antimonotoncity*, a fairly obvious property of sequence support is exploited.

Property 3. *(Antimonotone)*
Given a sequential database $\mathcal{S}_{\mathcal{DB}}$ *over* $\mathcal{I}$ *and two sequences s and s′. Then,*

$$s \subseteq s' \Rightarrow support(s, \mathcal{S}_{\mathcal{DB}}) \geqslant support(s', \mathcal{S}_{\mathcal{DB}}).$$

Hence, if a sequence is infrequent, all of its supresequences must be infrequent, and vice versa, if a sequence is frequent, all of its subseqeucnes must be frequent too. This *anti-monotonicity* property is also called the *downward closure* property, since the set of sequential patterns is downward closed with respect to sequence inclusion. The following property is considered to be as the central property for construction efficient algorithms for extraction sequential patterns:

Property 4. *(Apriori)*
Given a sequential database $\mathcal{S}_{\mathcal{DB}}$ over $\mathcal{I}$, two sequences s and s', and minimal support threshold σ_s. Then,

$$(s \text{ is not a sequential pattern}) \wedge (s \subseteq s') \Rightarrow s' \text{ is not a sequential pattern.}$$

1.4 Sequential Pattern Mining Methods

The sequential pattern mining problem was first addressed by Agrawal and Srikant [1995] and was defined as follows.

"Given a database of sequences, where each sequence consists of a list of transactions ordered by transaction time and each transaction is a set of items, sequential pattern mining is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data sequences that contain the pattern."

Garofalakis et al. [1999] described it as follows.

"Given a set of data sequences, the problem is to discover subsequences that are frequent, that is, the percentage of data sequences containing them exceeds a user-specified minimum support."

Masseglia et al. [2003] described it as follows.

"The discovery of temporal relations between facts embedded in a database"

Zaki [2001] presented it as a process to

"Discover a set of attributes, shared across time among a large number of objects in a given database."

Generally, mining sequential pattern can be considered as a problem of sequence enumeration with respect to frequency constraint (support). This problem is quite similar to frequent itemsets mining from transactional database $\mathcal{T}_{DB}$ where the size of the search space is $2^{|I|}$, where $|I|$ is the number of different items present in $\mathcal{T}_{DB}$. In case of sequences, taking into account the order relation between itemsets involves a search space even larger than the issue of frequent itemsets search. For instance, if we have a sequence $s = \langle s_1 \cdots s_l \rangle$, the number of all potential sequences will be $2^{|s_1|+|s_2|\cdots+|s_l|}$.

A naive method for pattern extraction would generate all possible patterns and count their supports in the database. Unfortunately, this approach can not be treated in a reasonable time. To solve this problem, Agrawal and Srikant [1995] emphasizes on using the property 4 for mining frequent sequences. This property states that the support decreases monotonically during extension of the sequence (often called the Apriori). Thus, if a sequence is considered not frequent, it is not necessary to generate its super-sequences as they are not frequent. Since this

work, many approaches have been proposed in recent years to improve the methods for sequential pattern mining. Among all these methods, the main difference lies in the way of navigating the search space (breadth-first or depth-first) and the type data structures used for indexing the sequential database which facilitates fast mining of frequent sequences. The sequential pattern mining algorithms can be classified into three broad categories:

- Horizontal sequence mining.
- Vertical sequence mining.
- Projection-Based sequence mining.

1.4.1 Horizontal sequence mining

GSP algorithm

GSP (Generalized Sequential Patterns) was the first algorithm proposed by Srikant and Agrawal [1996]. This is a level wise mining algorithm for mining sequential pattern. The authors define k-sequence as a sequence with k items, e.g., $\langle\{b\}\{a,c\}\rangle$ is a 3-sequence.

GSP algorithm passes over data multiple times. In each pass k, GSP generates a set C_k of all the candidate k-sequences, then it computes their support and keeps the frequent sequences in the set F_k. This set then becomes the seed set for the next pass k+1 to generate new potentially frequent patterns (i.e, C_{k+1}). Each sequence in C_{k+1} contains one more item as compared to the sequences in F_k. The algorithm terminates when no new sequential pattern is found in a pass (i.e, $F_k = \varnothing$) or no candidate sequence can be generated (i.e, $C_{k+1} = \varnothing$) (see Figure 1.2).

The generation of C_{k+1} is done by joining each two sequences s_1, s_2 in F_k. The joining is done as follows: if the subsequence obtained by removing first item of s_1 is the same as the subsequence obtained by removing the last item of s_2. The candidate sequence is then generated by extending s_1 with the last item in s_2 which is called *I-extension*. If last item in s_2 was a separate itemset, the candidate sequence is then generated by adding this item as a separate itemset at the end of s_1 which is called *S-extension*. For example, sequence $s_1 = \langle\{a,b\}\{c\}\rangle$ combines with $s_2 = \langle\{b\}\{c,d\}\rangle$ to generate $c = \langle\{a,b\}\{c,d\}\rangle$ (*I-extension*). When, the sequence $s_1 = \langle\{a,b\}\{c\}\rangle$ joins with $s_2 = \langle\{b\}\{c\}\{d\}\rangle$ to generate $c = \langle\{a,b\}\{c\}\{d\}\rangle$ (*S-extension*).

In order to reduce the number of sequences in C_k, candidate sequences are organized as a *hash-tree*. The nodes of the hash-tree either contain a list of sequences as a leaf node or a hash table as an interior node. To find which candidate sequences are included in a data sequence, GSP traverses the tree by applying a hash function to each item of the data sequence. When a leaf is reached, it contains potential candidate for the data sequence.

GSP also integrates with time constraints, such as *maximum gap* and *minimal gap*. These time constraints specify a gap between any two itemset (transaction) in the sequence. If the distance between two itemsets in a sequence is not in *maximum gap* and *minimal gap* then these two itemsets are not consecutive.

PSP algorithm

PSP (Prefix Tree for Sequential Pattern) algorithm Masseglia et al. [1998] is an improvement of GSP algorithm Srikant and Agrawal [1996]. The difference between GSP and PSP is in the organization of the candidate sequences. GSP uses *hash tables* at each intermediary node of the candidate tree, whereas the PSP algorithm organizes the candidates in a *prefix-tree* for efficient counting.

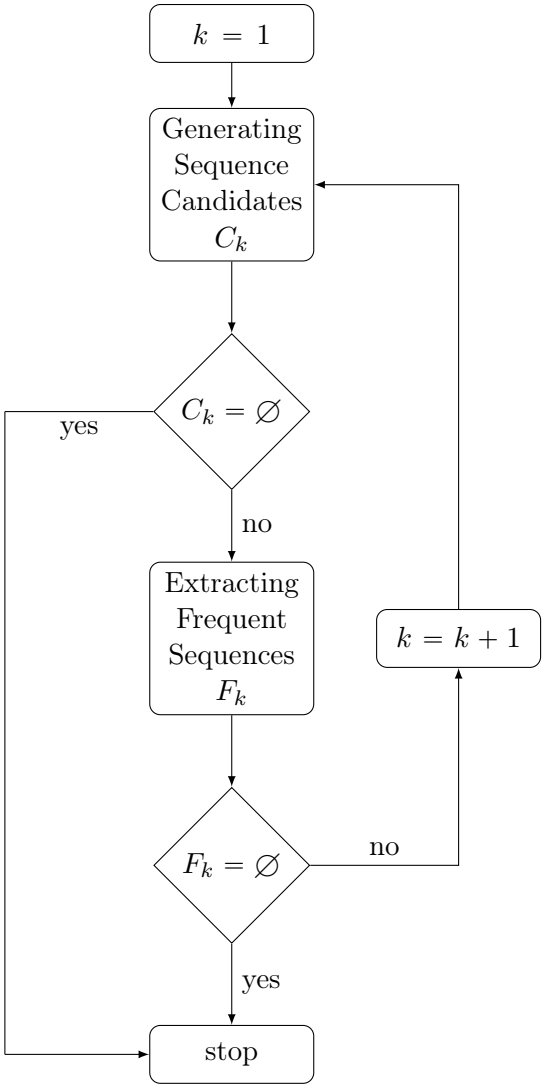


Figure 1.2: Diagram representing the GSP algorithm

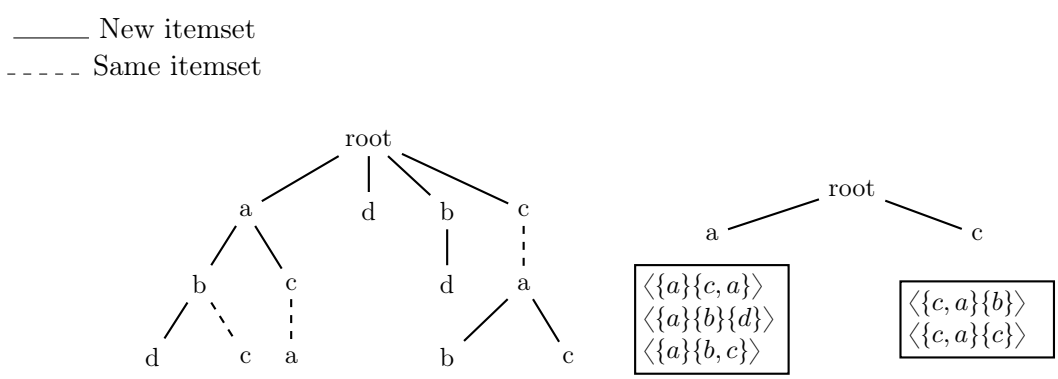


Figure 1.3: The *prefix-tree* of PSP (left tree) and the *hash-tree* of GSP (right tree) shows the difference between GSP and PSP for organizing of the candidate sequences $C_3 = \{\langle\{a\}\{c,a\}\rangle, \langle\{a\}\{b\}\{d\}\rangle, \langle\{a\}\{b,c\}\rangle, \langle\{c,a\}\{b\}\rangle, \langle\{c,a\}\{c\}\rangle\}$

In the *prefix-tree*, any branch from the root to the leaf represents one candidate. Each node of tree captures one item in the sequence. More precisely, PSP studies two cases during building the prefix-tree. If the two items are in the same itemset, dashed edge is used to join two items otherwise straight edge is used.

The purpose of using *prefix-tree* structure is to not store the candidate in its leaves as GSP. Figure 1.3 shows the difference between GSP and PSP for organizing the candidate sequences $C_3 = \{\langle\{a\}\{c,a\}\rangle, \langle\{a\}\{b\}\{d\}\rangle, \langle\{a\}\{b,c\}\rangle, \langle\{c,a\}\{b\}\rangle, \langle\{c,a\}\{c\}\rangle\}$. Figure 1.3 shows us the reducing overhead of the PSP algorithm.

MFS algorithm

MFS (Maximal Frequent Sequence) Zhang et al. [2001] is an algorithm for mining sequential patterns based on GSP algorithm. MFS is a modification of GSP ensuring smaller I/O cost compared with GSP. GSP scans database a number of times depending on the length of longest frequent sequences in the database. MFS first computes a rough estimate of the set of all frequent sequences by using previous mining results if the database is regularly updated and the frequent sequences are mined periodically. Otherwise, MFS mines a small sample of the database by using GSP to obtain rough estimate set. Then, this set is used to generate set of candidates which are checked against the database to determine which are in fact frequent. The maximal of these frequent sequences are kept and the process is repeated only on these maximal sequences checking any candidates against the database. The process terminates when no more new frequent sequences are discovered in an iteration. A major source of efficiency over GSP is that the supports of longer sequences can be checked earlier in the process.

MSPS algorithm

MSPS (Maximal Sequential Patterns using Sampling) Luo and Chung [2008] is a new algorithm for mining maximal sequential patterns by using multiple samples to effectively exclude infrequent candidates. MSPS adopts the candidate generation method in GSP Srikant and Agrawal [1996]. GSP counts all candidates on the whole database at each pass while MSPS tries to find and remove most of the infrequent candidates by counting as few candidates as possible. MSPS tries to count the candidates on a random sample drawn from the database at each pass to

$\langle\{a\}\rangle$	
Seq ID	Trans ID
1	1
2	1
3	1
4	1

$\langle\{c\}\rangle$	
Seq ID	Trans ID
2	2
3	1
4	2

$\langle\{e\}\rangle$	
Seq ID	Trans ID
1	1
4	3
5	1

$\langle\{b\}\rangle$	
Seq ID	Trans ID
2	2
4	2

$\langle\{d\}\rangle$	
Seq ID	Trans ID
2	2
4	2

Table 1.4: A Vertical Id-List

estimate which candidates are most potentially infrequent, which are then removed. Then, those candidates are removed which are infrequent w.r.t the original database.

MSPS uses a *prefix-tree* to store the candidates as PSP Masseglia et al. [1998]. The difference between PSP's tree and the one build by MSPS is that: (1) the MSPS *prefix-tree* is used to count candidates of different size, whereas PSP's *prefix-tree* is only used to count the candidate of the same size and (2) a bit vector is associated with MSPS *prefix-tree* to facilitate the sequence trimming. The size of bit vector is the number of distinct single items in the database where each bit represents an item and it is set to zero if the item is not frequent otherwise set to one. This vector assists in building the *prefix-tree*. Each sequence inserted into the tree is checked against the bit vector first and those items for which the bit vector is set to zero are trimmed accordingly. Afterwards, the *prefix-tree* is built for all the candidates. Finally, all the frequent sequences are obtained with help of top-down search.

1.4.2 Vertical Sequence Mining

The SPADE algorithm

SPADE (Sequential Pattern Discovery using Equivalence classes) Zaki [2001] is the first algorithm to use the vertical database format for extracting sequential patterns more effectively. This approach is an extension of the algorithm CHARM Zaki and Hsiao [2002] which focuses on the extraction of frequent itemsets. SPADE uses *a vertical id-list database format*, where they associate with each sequence a list of objects in which it occurs, along with the timestamps.

Example 5. *Table 1.5 shows the vertical format of the sequential dataset in Table 1.3.a. The support of a sequence is the cardinality of the set consisting of the sequence identifiers. For example, for the sequence $\langle\{a\}\rangle$, the support count is 4, it occurs with Seq ID 1, 2, 3 and 4. By scanning the vertical database, frequent 1-sequences can be generated with the minimum support.*

The authors use a lattice-theoretic approach to decompose the original search space (lattice) into smaller pieces (sub-lattices) according to equivalence classes built on an equivalence relation taking into account the prefix sequence. Thus, these equivalence classes are used to decompose the original problem into sub-problems which can be processed independently in main-memory. The equivalence classes of SPADE algorithm are defined using an equivalence relation on the prefix, i.e., two k-sequences are in the same class iff they share a common (k-1)-prefix. More formally, an equivalence class is defined as follows:

$$[s_1 \in F_{k-1}] = \{s_2 \in F_k; \mathcal{P}ref_{k-1}(s_2) = s_1\}$$

where, $\mathcal{P}ref_k(s)$ is the prefix of s until the length k. For example, the equivalence class of $\langle\{b\}\{a\}\rangle$ is :

$$[\langle\{b\}\{a\}\rangle] = \{\langle\{b\}\{a,b\}\rangle, \langle\{b\}\{a,d\}\rangle, \langle\{b\}\{a\}\{a\}\rangle, \langle\{b\}\{a\}\{d\}\rangle, \langle\{b\}\{a\}\{f\}\rangle\}$$

Seq ID	(Item,Trans ID) paris
1	$\langle(a,1)(e,2)\rangle$
2	$\langle(a,1)(b,2)(c,2)(d,2)\rangle$
3	$\langle(a,1)(c,1)\rangle$
4	$\langle(a,1)(b,2)(c,2)(d,2)(e,3)\rangle$
5	$\langle(e,1)\rangle$

Table 1.5: Vertical to horizontal database

SPEAD is a level wise mining algorithm, where at each new level, SPADE decomposes the sequences into equivalent classes. Then it enumerates all other frequent sequences from the sequences in each equivalence class without scaning the database. The enumeration is performed by joining the id-lists in one of three ways as follows:

Considering the three itemsets a, b and c and the sequence $s = \langle s_1, ..., s_l\rangle$, the joining is performed as,

- Joining a sequence made of a single itemset with another one made also of a single itemset results in a sequence made of a single itemset that is the union of the two previous itemsets. For example, joining $\langle a \cup c\rangle$ and $\langle b \cup c\rangle$ results in the sequence $\langle a \cup b \cup c\rangle$.

- Joining a sequence has one itemset with another one has two itemsets; i.e., joining $\langle a \cup c\rangle$ with $\langle b, c\rangle$, then the only possible outcome is new sequence $\langle b, a \cup c\rangle$.

- Joining two sequences are different only in the first itemset; i.e., joining $\langle a, s_1, \cdots, s_l\rangle$ with $\langle b, s_1, \cdots, s_l\rangle$, then there are three possible outcomes: $\langle a \cup b, s_1, \cdots, s_l\}\rangle$, $\langle a, b, s_1, \cdots, s_l\}\rangle$ and $\langle b, a, s_1, \cdots, s_l\}\rangle$. One special case occurs when $\langle a, s_1, \cdots, s_l\rangle$ is joined with itself which results in $\langle a, a, s_1, \cdots, s_l\rangle$.

Actually, SPADE scans the database only two times. In the first scan, *vertical id-list database format* is built and the frequent 1-sequences, F_1, is generated. For generating the frequent 2-sequences; i.e., F_2, SPADE performs a vertical-to-horizontal transformation as follows: SPADE scans its id-list and transform each (Seq ID , Trans ID) pair in the id-list of the item i into (i,eid) and insert it in the input-sequence sid. The result of conversion from vertical to horizontal database is shown in Table 1.5. Computing F_2 is done by scanning the recovered horizontal database (the second scan). SPADE forms a vertical id-list for all the sequences in F_2. Then, SPADE recursively decomposes the sequences at each new level into equivalence classes and all new frequent sequences can be enumerated by joining the id-lists of the sequences in each equivalence class.

The cSPADE algorithm

The cSPADE (constraint Sequential Pattern Discovery using Equivalence classes) is an extended version of SPADE Zaki [2000a]. cSPADE extends SPADE by introducing one or more constraint during the mining process. Following are the constraints used by cSPADE:

Length or width restrictions: In order to mine the database, cSPADE defines a restriction on length and number of patterns extracted. These constraints help cSPADE in avoiding the exponential explosion in the number of extracted sequential patters.

Seq ID	Trans ID	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$
1	1	1	0	0	0	0
1	2	0	0	0	0	1
2	1	1	0	0	0	0
2	2	0	1	1	1	0
3	1	1	0	1	0	0
4	1	1	0	0	0	0
4	2	0	1	1	1	0
4	3	0	0	0	0	1
5	1	0	0	0	0	1

Figure 1.4: Bitmap Representation of the dataset in Table 1.3.

Minimum or maximum gap: cSPADE defines constraints on consecutive sequence itemsets, which enables cSPADE to discover patterns that occur after a certain minimum amount of time and no longer than a specified maximum time.

Handling item constraints: SPADE uses two techniques to mine sequential patterns, vertical format and equivalence classes. Along with these two techniques, cSPADE incorporates constraints on the items that can appear in a sequence. Exclusion of an item becomes a simple check for a particular item in the class and the removal from those classes where it occurs. Further expansion of these classes would not contain these items.

Handling classes: This kind of constraint is only applicable for certain dataset where each input sequence has a class.

The SPAM algorithm

SPAM (Sequential PAttern Mining) algorithm, proposed by Ayres et al. [2002], uses a *depth-first* search with several pruning mechanisms. It also uses a *vertical bitmap* representation to store each sequence to count the support of the sequence efficiently. When SPAM scans the database for the first time, it constructs a vertical bitmap for each item. Each bitmap has a bit corresponding to transaction in the sequential database. If an item appears in the transaction, the bit corresponding to that transaction of the bitmap for the item is set to 1; otherwise, the bit is set to 0. Figure 1.4 shows the bitmap representation of the sequential database in Table 1.3.

All the candidate sequences are stored in a *prefix-tree* as in PSP Masseglia et al. [1998]. SPAM uses two steps to generate and test the candidate sequence: Sequence Extended (*S-extension*) and Itemset Extended (*I-extension*). *S-extension* appends an item at the end of the sequence as a new itemset while (*I-extension*) appends an item to the last itemset in the sequence.

In *S-extension*, each bitmap partition of a sequence to be extended is transformed, such that all bits after the first bit with value one in the partition are set to one and the first bit with value one is set to zero. Then the resultant bitmap of the *S-extension* is obtained by applying *AND* operation on the transformed bitmap and the bitmap of the appended item. On the other hand, the *I-extension* just uses bitmaps of the sequence and the appended item to perform *AND* operation to obtain the resultant bitmap. Figure 1.5 shows an example of *S-extension* process by extending the sequence $\langle\{a\}\rangle$ with the item $\{b\}$ and another example of *I-extension* process by extending the sequence $\langle\{a\}\{b\}\rangle$ with $\{d\}$. Then, SPAM uses depth-first search strategy

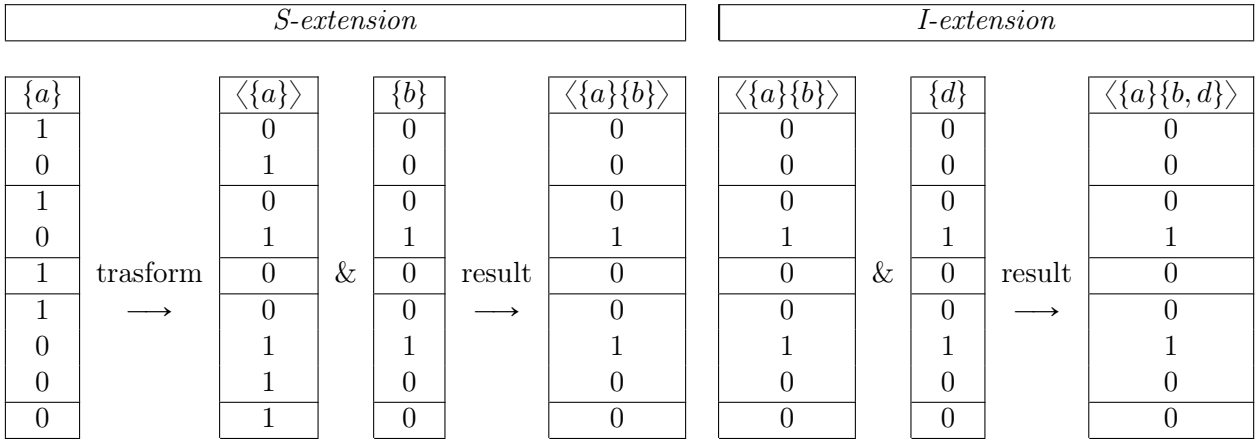


Figure 1.5: An example of *S-extension* and *I-extension*

for traversing the *prefix-tree*. The bitmap representation idea of SPAM requires quite a lot of memory, so it is very efficient for those databases which have very long sequential patterns.

The CCSM algorithm

The CCSM (Cache-based Constrained Sequence Miner) algorithm Orlando et al. [2004] focuses on mining sequential patterns. CCSM searches for sequential patterns level-wise and adopts an intersection-based method to determine the support of candidate k-sequences by using an effective cache which store intermediate id-lists. The similarity between CCSM and GSP Srikant and Agrawal [1996] is that CCSM adopts a level-wise bottom-up approach in visiting sequential patterns in the tree but it differs after the extraction of frequent F_1 and F_2 from the horizontal database. This pruned database is transformed into a vertical database resulting in the same configuration as SPADE Zaki [2001]. The difference between CCSM and SPADE is that CCSM uses cache to store intermediate id-lists for fast calculation of support. In order to build the id-list of another candidate, CCSM reuses the id-list corresponding to the common subsequence of this candidate if a common prefix of this candidate is already present in the cache. This way CCSM reduces the number of join operations to be performed. In addition, CCSM is also able to deal with very complex constraints, like the maximum temporal gap between itemsets occurring in the input sequences.

The LAPIN-SPAM algorithm

The LAPIN-SPAM (LAst Position INduction Sequential PAttern Mining) Yang and Kitsuregawa [2005] is proposed an approach which is based on the same principle as SPAM Ayres et al. [2002]. Where SPAM does many *AND* operations to generate candidate sequences. LAPIN-SPAM avoids this *AND* operation through the observation that if the last position of item a is smaller than, or equal to, the position of the last item in a sequence s, then the item can not appear behind the last item in a sequence during extending the sequence. Thus, LAPIN-SPAM avoids *AND* operation in each iteration in the process of support counting, which can largely improve the efficiency.

1.4.3 Projection-Based Sequence Mining

The FreeSpan algorithm

FreeSpan (**Fre**qu**e**nt pattern projected **S**equential **pa**ttern mining) is an algorithm introduced in Han et al. [2000a] for the reduction of candidate generation and testing in level-wise sequential pattern mining methods Srikant and Agrawal [1996]. *FreeSpan* uses the frequent items to recursively projected sequence database into a set of smaller projected databases. Thus, the execution time is improved because each projected database is smaller and easier to treat. The authors defined the projection of sequential database with an itemset or a sequence as follows:

Definition 21. *(x-projected sequential database)*
Let x be an itemset. The x-projected sequential database is defined as the collection of sequences having all items in x.

Definition 22. *(s-projected sequential database)*
Let s be a sequence, then s-projected sequential database is defined as the collection of sequences having s as subsequence.

Given the sequential database $\mathcal{S}_{DB}$ and minimum support threshold. As a first step, *FreeSpan* scans $\mathcal{S}_{DB}$ and extracts the set of frequent items $f_{list} = \{x_1, x_2, \cdots, x_n\}$, where the frequent items are sorted w.r.t to support in descending order. The items in f_{list} form 1-frequent sequential patterns F_1. According to the f_{list}, the complete set of sequential patterns in $\mathcal{S}_{DB}$ can be divided into n disjoint subsets: (1) the set of sequential patterns containing only item x_1, (2) those containing item x_2 but no item in $\{x_3, \cdots, x_n\}$, and so on. In general, the ith subset $(1 \leqslant i \leqslant n)$ is the set of sequential patterns containing item x_i but no item in $\{x_{i+1}, \cdots, x_n\}$.

The sequential patterns related to the n partitioned subsets can be mined by constructing n projected databases based on the n partitions of f_{list} where the infrequent items are removed from the projected databases. Mining each projected part extracts 2-sequential patterns. Then, *FreeSpan* projects each part of sequential database recursively into a set of small projected sequence databases based on the currently mined sequential patterns, and the new sequential patterns can be mined similarly on their corresponding projected databases.

This algorithm has provided a starting point for other studies on the projection of databases and sequential pattern mining. One of the limitations of the current algorithm is that it generates many nontrivial projected databases. Another shortcoming is that it keeps all the sequences in the projected databases without any reduction in its size. This problem has been solved in PrefixSpan Pei et al. [2001].

The *PrefixSpan* algorithm

The method PrefixSpan (**Prefix**-projected **S**equential **pa**ttern mining) proposed by Pei et al. [2001] is based on the idea of sequential database projection, presented in Han et al. [2000a], i.e., *FreeSpan* method. Major shortcoming of *FreeSpan* is that it generates many nontrivial projected databases. Moreover, the size of the projected database does not reduce if a pattern appears in each sequence in a database. The added advantage of *PrefixSpan* over *FreeSpan* is that it extends *FreeSpan* by taking into account the examination of prefix subsequences and projection of their corresponding postfix subsequences into projected databases. The concept of prefix and suffix is defined as follows:

Definition 23. *(Prefix of Sequence)*

Prefix	Projected database (*PrefixSpan*)	Projected database (*FreeSpan*)
$\langle\{a\}\rangle$	$\langle\{e\}\rangle$,$\langle\{b,c,d\}\rangle$,$\langle\{_,c\}\rangle$,$\langle\{b,c,d\}\{e\}\rangle$	$\langle\{a\}\rangle$,$\langle\{a\}\rangle$, $\langle\{a\}\rangle$,$\langle\{a\}\rangle$
$\langle\{c\}\rangle$	$\langle\{_,d\}\rangle$,$\langle\{_,d\}\{e\}\rangle$	$\langle\{a\}\{c\}\rangle$, $\langle\{a\}\{c\}\rangle$, $\langle\{a,c\}\rangle$ $\langle\{a\}\{c\}\rangle$
$\langle\{e\}\rangle$		$\langle\{a\}\{e\}\rangle$, $\langle\{a\}\{c\}\{e\}\rangle$, $\langle\{e\}\rangle$
$\langle\{b\}\rangle$	$\langle\{_,c,d\}\rangle$,$\langle\{_,c,d\}\{e\}\rangle$	$\langle\{a\}\{b,c\}\rangle$, $\langle\{a\}\{b,c\}\{e\}\rangle$
$\langle\{d\}\rangle$	$\langle\{e\}\rangle$	$\langle\{a\}\{b,c,d\}\rangle$, $\langle\{a\}\{b,c,d\}\{e\}\rangle$

Table 1.6: *FreeSpan* vesrus *PrefixSpan*

Suppose all the items within an itemset are listed alphabetically. Given a sequence $s = \langle s_1 s_2 \cdots s_n\rangle$ *(where each* s_i *corresponds to a frequent itemset in* s*), a sequence* $s^{'} = \langle s_1^{'} s_1^{'} \cdots s_m^{'}\rangle$ $(m \leqslant n)$ *is called a prefix of* s *if and only if: (1)* $s_i^{'} = s_i$ *for* $(i \leqslant m-1)$*; (2)* $s_m^{'} \subseteq s_m$*; and (3) all thee frequent items in* $(s_m - s_m^{'})$ *are alphabetically after those in* $s_m^{'}$*.*

Definition 24. *(Suffix of Sequence)*

Given a sequence $s = \langle s_1, s_2 \cdots s_n\rangle$ *(where each* s_i *corresponds to a frequent itemset in* s*). Let* $s^{'} = \langle s_1^{'} s_1^{'} \cdots s_m^{'}\rangle$ $(m \leqslant n)$ *be the prefix of* s*. Sequence* $s^{''} = \langle s_1^{''} s_2^{''} \cdots s_k^{''}\rangle$ *is called the suffix of* s *with regards to prefix* $s^{'}$*, denoted as* $s^{''} = s/s^{'}$*, if and only if* $\exists i \leqslant n; s_1^{''} = (s_i - s_m^{'})$ *and* $\forall 1 < j \leqslant k; (s_j^{''} = s_{i+j-1})$ *.*

Example 6. *Consider a sequence* $s = \langle\{a\}\{a,b,c\}\{a,c\}\{d\}\{c,f\}\rangle$*. The sequences* $\langle\{a\}\rangle$*,* $\langle\{a\}\{a\}\rangle$*,* $\langle\{a\}\{a,b\}\rangle$ *and* $\langle\{a\}\{a,b,c\}\rangle$ *are prefix of* s*. The sequence* $\langle\{a,b,c\}\{a,c\}\{d\}\{c,f\}\rangle$ *is the suffix of* s *with regards to prefix* $\langle\{a\}\rangle$ *while* $\langle\{_,b,c\}\{a,c\}\{d\}\{c,f\}\rangle$ *is the suffix of* s *with regards to prefix* $\langle\{a\}\{a\}\rangle$ *.*

Accordingly, the projected database is defined as follows:

Definition 25. *(s-projected database)*

Let s *be a sequential pattern in a sequential database* $\mathcal{S}_{DB}$*. The s-projected database, denoted as* $\mathcal{S}_{DB}|_s$ *is the collection of suffixes of sequences in* $\mathcal{S}_{DB}$ *with regards to prefix* s*.*

Example 7. *Table 1.6 demonstrates the difference between the databases projected by* FreeSpan *and* PrefixSpan *based on database shown in Table 1.3.a.*

PrefixSpan resembles *FreeSpan* because of the fact that it recursively projects a sequence database into a set of smaller projected sequence databases and grows sequential patterns in each projected database by exploring only locally frequent fragments. PrefixSpan applies two projection techniques (i) *bi-level projection* and (ii) *pseudo projection* Pei et al. [2004b] for reducing the number and the size of projected databases.

In *bi-level projection*, during each step *PrefixSpan* builds an $n \times n$ triangle matrix *S-matrix*. This matrix represents the support of those sequences which are created by each two frequent items obtained from the projected database at each step. For example, *S-matrix*[i,j]=(a,b,c) where *i,j* are the indices, where a, b, c are the support of the sequences $\langle\{i\}\{j\}\rangle$, $\langle\{i,j\}\rangle$ and $\langle\{j\}\{i\}\rangle$ respectively. The construction of the triangle matrix *S-matrix* is repeated iteratively for each k-projected database.

The *physical projection* may lead to repeated copying of different suffixes of the sequence in several projected databases. This problem of redundancy is overcome with the help of *pseudo-projection*. This type of projection represents each sequence in the projected database by a corresponding projection position (an index point) instead of copying the whole suffix as a

Prefix	Projected (postfix) database	2-Sequential patterns
$\langle\{a\}\rangle$	$\langle\{e\}\rangle$,$\langle\{b,c,d\}\rangle$,$\langle\{_,c\}\rangle$,$\langle\{b,c,d\}\{e\}\rangle$	$\langle\{a\}\{b\}\rangle$,$\langle\{a\}\{c\}\rangle$, $\langle\{a\}\{d\}\rangle$,$\langle\{a\}\{e\}\rangle$
$\langle\{b\}\rangle$	$\langle\{_,c,d\}\rangle$,$\langle\{_,c,d\}\{e\}\rangle$	$\langle\{b,c\}\rangle$, $\langle\{b,d\}\rangle$
$\langle\{c\}\rangle$	$\langle\{_,d\}\rangle$,$\langle\{_,d\}\{e\}\rangle$	$\langle\{c,d\}\rangle$
$\langle\{d\}\rangle$	$\langle\{e\}\rangle$	

Table 1.7: Level-by-Level Projected Database to generate frequent 2-sequences

$\{a\}$	0				
$\{b\}$	(0,0,1)	0			
$\{c\}$	(0,1,2)	(0,2,0)	0		
$\{d\}$	(0,0,2)	(0,2,0)	(0,2,0)	0	
$\{e\}$	(0,0,2)	(0,0,1)	(0,0,1)	(0,0,1)	0
	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$

Table 1.8: The *S-matrix* matrix

projected subsequence. An index position pointer may save physical projection of the suffix and, thus, save both space and time of generating numerous physical projected databases. Thus, *pseudo-projection* avoids physically copying suffixes.

Example 8. *This example illustrates the application of PrefixSpan on the sequential database (see Table 1.3) with minimum support threshold equals to 2. First, the database is scanned and frequent 1-sequences are generated. Then, the database is projected according to these frequent sequences. Table 1.7 shows the projected database generated according to the frequent 1-sequences. After that, a triangular matrix S-matrix is constructed. Table 1.8 shows the triangle matrix S-matrix constructed after scanning the projected databases in Table 1.7 for computing the support of 2-sequences. For instance, S-matrix[{c},{a}]=(0,1,2) means that support of* $\langle\{c\}\{a\}\rangle$, $\langle\{a,c\}\rangle$ *and* $\langle\{a\}\{c\}\rangle$ *is 0,1 and 2 respectively. The frequent 2-sequences in Table 1.7 are generated depending on the values of the support in S-matrix (see Table 1.8). The triangle matrix S-matrix and projected database are constructed iteratively for each frequent k-sequence until the projected database becomes empty or no frequent sequence can be found. The* pseudo-projection *for prefixes* $\langle\{a\}\rangle$, $\langle\{c\}\rangle$, $\langle\{e\}\rangle$, $\langle\{b\}\rangle$ *and* $\langle\{d\}\rangle$ *are shown in Table 1.9, where $ indicates that the prefix occurs in the current sequence but its projected suffix is empty, whereas* $\varnothing$ *indicates that there is no occurrence of the prefix in the corresponding sequence.*

Sequence id	sequence	$\langle\{a\}\rangle$	$\langle\{c\}\rangle$	$\langle\{e\}\rangle$	$\langle\{b\}\rangle$	$\langle\{d\}\rangle$
1	$\langle\{a\}\{e\}\rangle$	2	$\varnothing$	\$	$\varnothing$	$\varnothing$
2	$\langle\{a\}\{b,c,d\}\rangle$	2	4	$\varnothing$	3	\$
3	$\langle\{a,c\}\rangle$	2	\$	$\varnothing$	$\varnothing$	$\varnothing$
4	$\langle\{a\}\{b,c,d\}\{e\}\rangle$	2	4	\$	3	5
5	$\langle\{e\}\rangle$	$\varnothing$	$\varnothing$	\$	$\varnothing$	$\varnothing$

Table 1.9: A Sequence Database in Table 1.3.a and Some of Its Pseudoprojected Databases

1.4.4 Extension

Closed Sequential Patterns

Yan et al. [2003] emphasizes on the use of closed sequential patterns for concise representation of sequential patterns. It uses a transactional database containing only one sequence: $\langle\{a_1\}\{a_2\}\cdots\{a_{100}\}\rangle$. If this sequence is mined with a minimum support threshold of one, classic sequential pattern mining methods presented before generate $2^{100}-1$ sequential patterns. All the patterns are redundant because they have the same support as that of $\langle\{a_1\}\{a_2\}\cdots\{a_{100}\}\rangle$. A similar problem is encountered while mining frequent itemset. In order to overcome this problem, Pasquier et al. [1999] has proposed an interesting solution called *mining frequent closed itemsets*. There are several other algorithms for mining frequent closed itemset such as *CLOSET* Pei et al. [2000], *CHARM* Zaki and Hsiao [2002] and *CLOSET+* Wang et al. [2003]. In sequence mining, problem of closed sequential patterns is defined by Yan et al. [2003] as follows:

Definition 26. *Given a sequential database $\mathcal{S}_{\mathcal{DB}}$ and a minimum support threshold σ_s. The set of closed sequential pattern, denoted as $\mathcal{CSP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s)$, is defined as follows,*

$$\mathcal{CSP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s) = \{s_1; s_1 \in \mathcal{SP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s) \wedge \nexists s_2; s_1 \preceq s_2 \text{ and } support(s_1, \mathcal{S}_{\mathcal{DB}}) = support(s_2, \mathcal{S}_{\mathcal{DB}})\}$$

Example 9. *In our sequential database in Table 1.3, for a minimum support threshold equals to 2, the set of the sequential patterns $\mathcal{SP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s)$ has 17 patterns while the set of closed sequential patterns $\mathcal{CSP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s)$ has 5 closed patterns.*

The set of sequential patterns extracted is:

$$\begin{aligned}\mathcal{SP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s) \quad = \quad & \{\langle\{a\}\rangle : 4,\ \langle\{c\}\rangle : 3,\ \langle\{e\}\rangle : 3,\ \langle\{a\}\{b\}\rangle : 2,\ \langle\{a\}\{b,c\}\rangle : 2,\ \langle\{a\}\{b,c,d\}\rangle : 2,\\ & \langle\{a\}\{d\}\rangle : 2,\ \langle\{a\}\{c\}\rangle : 2,\ \langle\{a\}\{d\}\rangle : 2,\ \langle\{a\}\{c,d\}\rangle : 2,\ \langle\{a\}\{e\}\rangle : 2,\ \langle\{b\}\rangle : 2,\\ & \langle\{b,c\}\rangle : 2,\ \langle\{b,c,d\}\rangle : 2,\ \langle\{b,d\}\rangle : 2,\ \langle\{c,d\}\rangle : 2,\ \langle\{d\}\rangle : 2\}\end{aligned}$$

While the set of closed sequential patterns is:

$$\mathcal{CSP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s) = \{\langle\{a\}\rangle : 4,\ \langle\{c\}\rangle : 3,\ \langle\{e\}\rangle : 3,\ \langle\{a\}\{b,c,d\}\rangle : 2,\ \langle\{a\}\{e\}\rangle : 2\}$$

To date there have been several studies for the extraction of closed itemsets Bayardo [1998], Pasquier et al. [1999], Pei et al. [2000], Zaki and Hsiao [2002], Wang et al. [2003], Grahne and Zhu [2005], Liu et al. [2003], Pan et al. [2003] and Borgelt et al. [2011], there is, to our knowledge, only three proposals for closed sequential patterns such as CloSpan Yan et al. [2003], BIDE Wang and Han [2004] and ClaSP Gomariz et al. [2013]. CloSpan and BIDE both are derived from PrefixSpan Pei et al. [2001] while ClaSP uses vertical database format Zaki [2001].

I. The CloSpan algorithm

Yan et al. [2003] proposed a *CloSpan* (**Clo**sed **S**equential **pa**ttern**n** mining) algorithm for extracting closed sequential patterns. *CloSpan* divides the task into two steps. In the first step, the algorithm generates a set of closed sequence candidates. The generated candidate set is usually larger than the final closed sequences set. The second steps removes all non-closed sequences.

To extract the closed sequence candidates the authors firstly defines a lexicographic order between two sequences as follows. Given two sequences s_1 and s_2 and two itemsets i and i'. The sequence s_1 is discovered after s_2, denoted by $s_1 < s_2$, if and only of:

- $s_2 = s_1 \diamond i$; $\diamond$ is either *I-extension* or *S-extension*, or

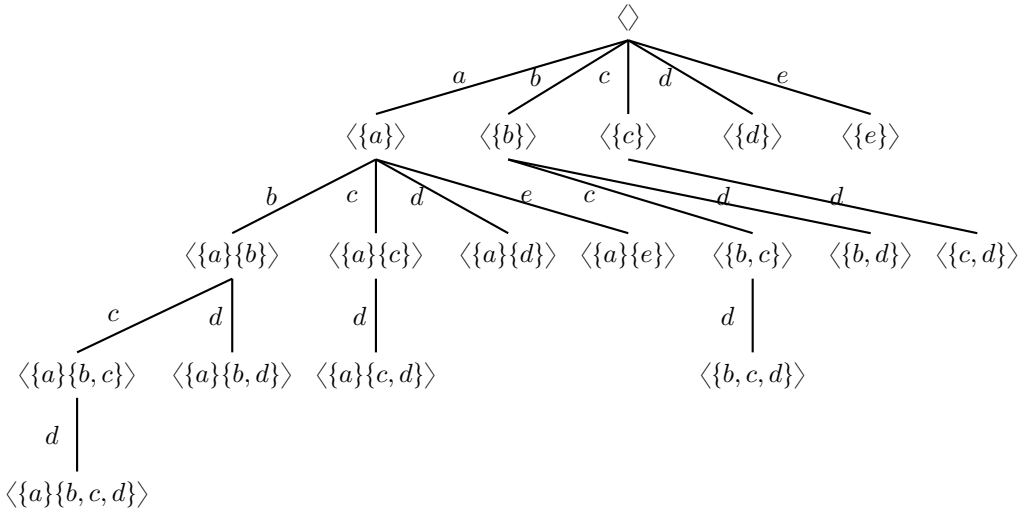


Figure 1.6: Lexicographic sequence tree for the sequential database in Table 1.3.a

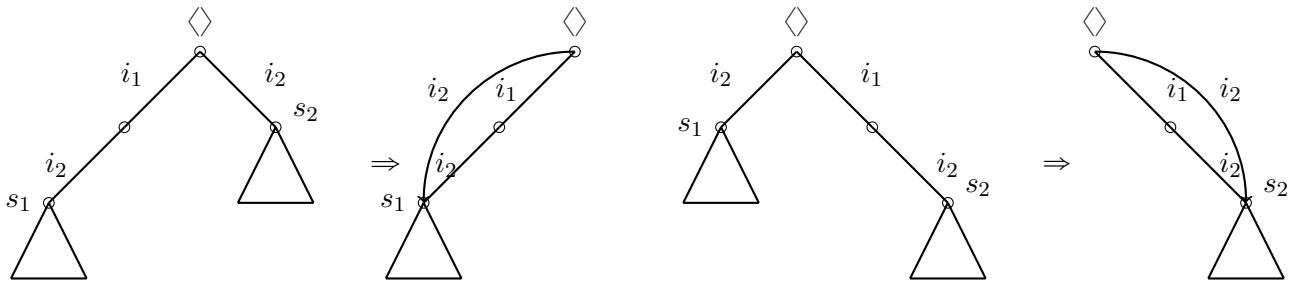
- $s_1 = s_3 \diamond_i i$ and $s_2 = s_3 \diamond_s i^{'}$; $\diamond_i$ is *I-extension* and $\diamond_s$ is *S-extension*, or
- $s_1 = s_3 \diamond i$ and $s_2 = s_3 \diamond i^{'}$; $\diamond$ is either *I-extension* or *S-extension* and i comes before $i^{'}$ in the item lexicographic order

For example $\langle\{a, b\}\rangle < \langle\{a, b\}\{a\}\rangle$ as $\langle\{a, b\}\{a\}\rangle = \langle\{a, b\}\rangle \diamond_s \{a\}$. Another example $\langle\{a, b\}\rangle < \langle\{a\}\{a\}\rangle$ as $\langle\{a, b\}\rangle = \langle\{a\}\rangle \diamond_i \{b\}$ and $\langle\{a\}\{a\}\rangle = \langle\{a\}\rangle \diamond_s \{a\}$.

CloSpan uses projection databases idea which is presented by Pei et al. [2001] for generating a set of closed sequence candidates The sequence candidates are stored in *lexicographic sequence tree* structure. In this structure, each node is a sequence where empty sequence is taken as the root. Given a sequence s in the tree, its child would either be an I-extension of s, or a S-extension of s. Moreover, the left sibling is less than the right sibling in sequence lexicographic order. Figure 1.6 shows an example of the lexicographic sequence tree for the frequent sequences extracted from Table 1.3.a.

There are two techniques used for pruning the *lexicographic sequence tree* which are: *backward sub-pattern* and *backward super-pattern*. The *backward sub-pattern* technique is based on the following observation: if we have sequence $s_1 < s_2$, $s_1 \supset s_2$ and the projection database by s_1 equals to the projection database by s_2, then there is no need to extend s_2 anymore since all the children of s_2 are the same as that of s_1 with the same support. While the *backward super-pattern* technique is based on the following observation: if we have sequence $s_1 < s_2$, $s_1 \subset s_2$ and also the projection database by s_1 equals to the projection database by s_2, then it is sufficient to transplanting the descendant of s_1 to s_2 instead of searching any descendant of s_2 in the *lexicographic sequence tree*. Figure 1.7 shows that the subtrees of s_1 and s_2 are merged into one without mining the subtree under s_2. With the help of *backward sub-pattern* and *backward super-pattern* techniques a *lexicographic sequence tree* can be replaced with a *lexicographic sequence lattice*.

The second step after generating the closed sequence candidates is removing all non-closed sequences from the *lexicographic sequence lattice*. The problem encountered is to check out for each sequence s_1, whether there exists a super-sequence s_2 s.t. $support(s_1, \mathcal{S}_{DB}) = support(s_2, \mathcal{S}_{DB})$. As a solution, *CloSpan* first searches for all the sequences with the same support as that of s,



(a) *backward sub-pattern* (a) *backward super-pattern*

Figure 1.7: Backward Sub-Pattern and Super-Pattern

then it checks whether there is a super-sequences containing s.

II. The BIDE algorithm

CloSpan encounters efficiency issues for a huge number of closed sequential patterns in the sequential database. To overcome this problem, Wang and Han [2004] proposed a new algorithm BIDE (**BI-D**irectional **E**xtension) to mine a huge number of closed sequential patterns effectively. BIDE avoids maintaining the set of closed sequence candidates by designing a new sequences closure checking method called *BI-Directional Extension checking* Wang et al. [2007].

BI-Directional Extension checking states that: *"If there exists no forward-extension nor backward-extension w.r.t. a prefix sequence s, s must be a closed sequence; otherwise, s must be non-closed.".*

The general idea underlying this method extends the sequence from both directions from front (*forward-extension*) and from back (*backward-extension*). The forward directional extension is used to grow all of the prefix patterns and check for closure of these patterns, while the backward directional extension is used to both check for closure and prune the search space.

III. The ClaSP algorithm

Gomariz et al. [2013] proposed a new algorithm, called ClaSP (Closed Sequential Pattern algorithm). This algorithm is used the same mechanism of CloSpan Yan et al. [2003] as method to mine close sequences and a vertical id-list database format to store each sequence to count the support of the sequence efficiently Zaki [2001]. ClaSP shows that under some database configurations, a vertical database format for mining closed sequential pattern can be faster than a horizontal database format for doing the same objective.

Approximate Sequential Patterns

Kum et al. [2003] proposes a new approach *ApproxMAP* (**Approx**imate **M**ultiple **A**lignment **P**attern mining) for the approximation of sequential pattern mining in databases consisting of long sequences. *ApproxMAP* is based on two steps cluster the sequences and then multiple alignment based approach.

In the first step, sequences in a database are clustered based on similarity. In order to cluster the database, the edit distance Xu and Zhang [2004] is used as a distance measure for sequence metric, then the density-based method is used for clustering the sequences Zhou et al. [1999].

In the second step, the algorithm generates longest approximate sequential pattern for each cluster called *consensus pattern*. Consensus sequential patterns are mined directly through a process of multiple alignment. During this process, in each partition two sequences are aligned,

ID	Sequence
s_1	$\langle\{b,c\}\{d,e\}\rangle$
s_2	$\langle\{a\}\{b,c,x\}\{d\}\rangle$
s_3	$\langle\{a,e\}\{b\}\{b,c\}\{d\}\rangle$
s_4	$\langle\{a\}\{b\}\{d,e\}\rangle$
s_5	$\langle\{a,x\}\{b\}\{b,c\}\{z\}\{a,e\}\rangle$

ID	Sequence
s_6	$\langle\{a,y\}\{b,d\}\{b\}\{e,y\}\rangle$
s_7	$\langle\{a,j\}\{p\}\{k\}\{l,m\}\rangle$
s_8	$\langle\{i,j\}\{k,q\}\{m\}\rangle$
s_9	$\langle\{i\}\{l,m\}\rangle$
s_{10}	$\langle\{v\}\{p,w\}\{e\}\rangle$

Table 1.10: Sequential Dataset

s_2	$\langle\{a\}$	$\{\}$	$\{b,c,x\}$	$\{\}$	$\{d\}\rangle$
s_3	$\langle\{a,e\}$	$\{b\}$	$\{b,c\}$	$\{\}$	$\{d\}\rangle$
s_4	$\langle\{a\}$	$\{\}$	$\{b\}$	$\{\}$	$\{d,e\}\rangle$
s_1	$\langle\{\}$	$\{\}$	$\{b,c\}$	$\{\}$	$\{d,e\}\rangle$
s_5	$\langle\{a,x\}$	$\{b\}$	$\{b,c\}$	$\{z\}$	$\{a,e\}\rangle$
s_6	$\langle\{a,y\}$	$\{b,d\}$	$\{b\}$	$\{\}$	$\{e,y\}\rangle$
s_{10}	$\langle\{v\}$	$\{\}$	$\{\}$	$\{p,w\}$	$\{e\}\rangle$
Weighted Sequence	{a:5,e:1, v:1, y:1}:6	{b:3, d:1}:3	{b:6, c:4, x:1}:6	{p:1, w:1, z:1}:2	{a:1,d:4, e:5, y:1}:7
Consensus Pattern	$\langle\{a\}$		$\{b,c\}$		$\{d,e\}\rangle$

Table 1.11: Cluster 1 (*min_ strenth=50 %* $\wedge$ $w \geqslant 4$)

then a sequence is added incrementally to the current alignment until all sequence have been aligned. A weighted sequence is used to summarize the alignment sequences in each cluster. An alignment sequence is a sequence of itemsets with a weight associated to each item. The weight of the item corresponds to the strength of the item, here strength refers to the number of sequences in the alignment having an item present in the aligned position. Consensus patterns are then selected based on the weighted sequence along with a strength threshold with the help of which those items are removed whose strength is less than the threshold.

Example 10. *Given a sequential dataset in Table 1.10, in the first step ApproxMAP generates two clusters of the data (see Tables 1.11 and 1.12). Then it aligns the sequences in each cluster by compressing all the sequences into one weighted sequence per cluster. Finally, it summarizes the weighted sequences into consensus patterns using the cut off point min strength. The patterns* $\langle\{a\}\{b,c\}\{d,e\}\rangle$ *and* $\langle\{i,j\}\{k\}\{l,m\}\rangle$ *are the consensus patterns extracted from two clusters in Table 1.11 and Table 1.12.*

The ProMFS (**Pro**babilistic **M**ining **F**requent **S**equences) proposed by Tumasonis and Dzemyda [2004] uses estimated probabilistic-statistical characteristics of the appearance of items of the sequence and their order: (1) the probability of an item in the sequence, (2) the probability for one item to appear after another one, and (3) the average distance between different items of the sequence Tumasonis and Dzemyda [2005]. This technique generates a smaller model sequence with the help of these estimated characteristics. It makes decision on the main sequence according to the analysis results of the shorter sequence. Afterwards the model sequence can be analyzed using GSP Srikant and Agrawal [1996] or other algorithm for mining sequential patterns.

s_8	$\langle\{i,j\}$	$\{\}$	$\{k,q\}$	$\{m\}\rangle$
s_7	$\langle\{a,j\}$	$\{p\}$	$\{k\}$	$\{l,m\}\rangle$
s_9	$\langle\{i\}$	$\{\}$	$\{\}$	$\{l,m\}\rangle$
Weighted Sequence	$\{a:1,i:2,j:2\}:3$	$\{p:1\}:1$	$\{k:2,q:1\}:2$	$\{l:2,m:3\}:3$
Consensus Pattern	$\langle\{i,j\}$		$\{k\}$	$\{l,m\}\rangle$

Table 1.12: Cluster 2 ($min_strenth$=50 % $\wedge$ $w \geqslant 2$)

Parallel Algorithms

All the previous algorithms for mining sequential patterns are serial algorithms. For finding sequential patterns a large amount of customer transaction data is to be searched which requires multiple passes over the database taking long computation time. Thus, this section discusses all the parallel algorithms for finding frequent sequences.

Guralnik and Karypis [2004] proposed three parallel algorithms for mining sequential patterns on a shared environments which are **N**on **P**artitioned **S**equential **P**attern **M**ining (*NPSPM*), **S**imply **P**artitioned **S**equential **P**attern **M**ining (*SPSPM*) and **H**ash **P**artitioned **S**equential **P**attern **M**ining (*HPSPM*). In *NPSPM*, each node stores a copy of sequences, it determines the frequent sequences by exchanging the support count values among all the nodes. These nodes can work independently in counting support. The sequences in this algorithm are copied instead of being partitioned. Often, the sequence becomes too large to fit within the local memory of a single node. In order to overcome this problem, *SPSPM* generates equal partitions of the sequences on all the nodes. For counting support, each node has to broadcast the sequences stored in its local disk to all the other nodes, while *NPSPM* does not require this broadcast. Due to this broadcast, the amount of communication in *SPSPM* becomes very large. To solve this problem, *HPSPM* partitions the sequences among the node using hash function to avoid the broadcast among all the nodes.

Demiriz [2002] proposed *webSPADE*, a parallel sequence mining algorithm to perform click stream analysis by modifying *SPADE* algorithm Zaki [2001]. Actually, The *SPADE* algorithm decomposes the search space of sequential patterns into independent classes, and thus is well-suited for parallel processing. The authors use a dynamic load- balancing scheme to ensure equal distribution of work among the processors. Parallelism is achieved easily by multi-threaded programming and using shared-memory. The authors developed this algorithm to analyze Web log data that had been cleaned and stored in a data warehouse.

Guralnik and Karypis [2004] developed two distributed algorithms which use projection-based methods for mining sequential patterns Pei et al. [2001]. The algorithms take advantage of either the *data-parallelism* or *task-parallelism*. The *data-parallelism* is achieved by partitioning the database across the different processors, whereas the *task-parallelism* formulation decomposes the computation by using task-decomposition to partition the lattice of frequent patterns. Both of these processes take advantage of dynamic load-balancing scheme Menon and Kalé [2013]. This way these algorithms reduce the data sharing overheads by minimizing the portions of the database that needs to be shared by different processors.

Par-CSP (**Par**allel **C**losed **S**equential **P**attern mining) is the first parallel algorithm for mining closed sequential patterns on a distributed memory system Cong et al. [2005]. It uses serial algorithm *BIDE* Wang and Han [2004] to mine the closed sequential patterns without candidate

cid	Cut-Grp	City	Age-grp	product-sequence
10	business	Boston	middle	$\langle\{b,d\}\{c\}\{b\}\{a\}\rangle$
20	professional	Chicago	young	$\langle\{b,f\}\{c,e\}\{f,g\}\rangle$
30	business	Chicago	middle	$\langle\{a,h\}\{a\}\{b\}\{f\}\rangle$
40	education	New York	retired	$\langle\{b,w\}\{c,e\}\rangle$

Table 1.13: A Multi-dimensional sequence database.

maintenance. The algorithm takes advantage of *divide-and-conquer* rule to minimize the inter-processor communications. *Par-CSP* uses dynamic scheduling for task assignment. Furthermore, it also uses *selective sampling* to estimate the relative mining time of the subtasks and to achieve load balancing.

1.5 Multi Dimensional and Multilevel Sequential Patterns Algorithms

1.5.1 Multi Dimensional Sequential Patterns Methods

In the previous section, we discussed that all the sequential pattern mining methods aim at extracting correlations within a single attribute in a relation of order Agrawal and Srikant [1995]. With the help of several attributes or dimensions we discover patterns that best describe the data Pinto et al. [2001]. For example, a consumer database can hold information such as article price, gender of the customer, location of the store and so on. If a customer often buys product a at $city_1$ then product b at $city_2$ one week later. The sequence $\langle(a, city_1)(b, city_2)\rangle$ is frequently supported by customer and may be more useful as compared to the sequence $\langle\{a\}\{b\}\rangle$.

Multidimensional sequence was defined for the first time by Pinto et al. [2001] as follows:

" A multidimensional sequential database is of schema $(RID, A_1, ..., A_n, S)$, where RID is a primary key, $A_1, ..., A_n$ are dimensions and S is in the domain of the sequences. Let $*$ be a meta-symbol which does not belong to any domain of $A_1, ..., A_n$. A multidimensional sequence takes the form if $(a_1, ..., a_n, s)$, where $a_i \in (A_i \cup \{*\}$ for $(1 \leqslant i \leqslant n)$ and s is a sequence."

Table 1.13 shows a sample multidimensional sequential database. Multidimensional sequential patterns mining aims at extracting the most interesting sequential patterns by considering several attributes or dimensions. The authors propose three algorithms *UniSeq*, *Dim-Seq* and *Seq-Dim* by using PrefixSpan as a sequential pattern mining algorithm Pei et al. [2001], and BUC-like (Bottom Up Computation) as a multidimensional pattern mining algorithm Beyer and Ramakrishnan [1999] as the basis.

UniSeq (Unique Sequence) is the first approach proposed by Pinto et al. [2001] where the multi-dimensional attributes are embedded in the sequential database by adding a new itemset in the sequence database. Afterwards it applies PrefixSpan on the newly defined sequential database in Table 1.14.

The other two approaches, *Dim-Seq* and *Seq-Dim*, partition a multidimensional sequential data set into two parts, first is the dimensional information while the second is the former sequences. Thus, multidimensional sequential pattern mining can be decomposed into two sub-problems: mining multidimensional patterns and mining sequential patterns. As a first step, BUC-like algorithm Beyer and Ramakrishnan [1999] is used to mine the multidimensional pat-

cid	MD-extension of sequence
10	$\langle\{business, Boston, middle\}\{b, d\}\{c\}\{b\}\{a\}\rangle$
20	$\langle\{professional, Chicago, young\}\{b, f\}\{c, e\}\{f, g\}\rangle$
30	$\langle\{business, Chicago, middle\}\{a, h\}\{a\}\{b\}\{f\}\rangle$
40	$\langle\{education, NewYork, retired\}\{b, w\}\{c, e\}\rangle$

Table 1.14: MD-extension of sequence.

terns. The frequent one dimensional items can be obtained after the first scan of the database. Similarly, the two dimensional frequent items are extracted and so on until dimensional frequent patterns no longer exist. Then, the sequential part of a multidimensional sequential data set is projected by using the frequent multidimensional patterns. PrefixSpan is used to mine sequential patterns in the projected sequential data sets. More precisely, in *Dim-Seq*, multidimensional patterns are extracted and for each of these patterns a sequential data set is projected to extract sequential patterns. While, in *Seq-Dim*, sequential patterns are first extracted from the sequential data set part, then for each sequential pattern, multidimensional pattern mining is applied on the projected multidimensional data set part.

Experiments were conducted to compare the three algorithms *UniSeq*, *Dim-Seq* and *Seq-Dim*. The results of these experiments show that *Seq-Dim* is the most efficient in most cases, but for less number of dimensions *UniSeq* algorithm is the fastest Pinto et al. [2001].

Yu and Chen [2005] proposed two algorithms for mining sequential patterns from the multidimensional sequential dataset, *AprioriMD* and *PrefixMDSpan* algorithms. For explaining the process, the authors introduced two different representations of multidimensional sequences: the standard format and the simplified format. In the standard format, a dimensional sequence would be represented in a natural way, i.e., as a d-dimensional structure. While, in the simplified format, a dimensional sequence would be represented as a linear structure. The authors used the standard format to define the problem and the simplified format to design the algorithms.

This study uses online stock-trading site as an example to illustrate the standard and simplified format of multidimensional sequences. In this data set, each customer may visit a Web site in a series of days; each day takes a series of sessions and each session visits a series of Web pages. Then, the data for each customer forms three dimensions, where the first dimension is days, the second is sessions, and the third is visited pages. For example, suppose we have a certain customer who visits the Web site in two successive days. On the first day, the customer traverses the site a and b in the first session and then d, c, and f pages in the second session. On the second day, the customer traverses the site d and f successively in the first session and then b and c successively in the second session. Thus, the standard format of this sequence can be expressed as $\langle\langle\langle ab\rangle_1\langle dcf\rangle_1\rangle_2\langle\langle df\rangle_1\langle bc\rangle_1\rangle_2\rangle_3$ where 1, 2, and 3 correspond to visited pages, sessions, and days, respectively. Then, the authors transferred the standard format of the sequence to a simplified format as follows. Each item e in the standard sequence is transferred to pair *(item,scop)* where *item* is the value of e and *scop* represents the immediate shared dimension between the current item e and the direct previous item of e. For example, the third item d in the previous example $\langle\langle\langle ab\rangle_1\langle dcf\rangle_1\rangle_2\langle\langle df\rangle_1\langle bc\rangle_1\rangle_2\rangle_3$ is transformed into (d,2) because the immediate shared dimension between d and the item b (which is directly before d) is the dimension 2. Thus, the simplest format of $\langle\langle\langle ab\rangle_1\langle dcf\rangle_1\rangle_2\langle\langle df\rangle_1\langle bc\rangle_1\rangle_2\rangle_3$ is the sequence $\langle(a,1)(b,1)(d,2)(c,1)(f,1)(d,3)(f,1)(b,2)(c,1)\rangle$. Table 1.15 shows a multidimensional sequential database represented in both formats.

After defining the two formats, the authors proposed two algorithms *AprioriMD* and *Pre-*

Sid	The standard format	The simplified format
1	$\langle\langle\langle ab\rangle_1\langle dcf\rangle_1\rangle_2\langle\langle df\rangle_1\langle bc\rangle_1\rangle_2\rangle_3$	$\langle(a,1)(b,1)(d,2)(c,1)(f,1)(d,3)(f,1)(b,2)(c,1)\rangle$
2	$\langle\langle\langle ab\rangle_1\rangle_2\langle\langle c\rangle_1\langle be\rangle_1\rangle_2\langle\langle g\rangle_1\langle hf\rangle_1\rangle_2\rangle_3$	$\langle(a,1)(b,1)(c,3)(b,2)(e,1)(g,1)(h,2)(f,1)\rangle$
3	$\langle\langle\langle bc\rangle_1\langle ac\rangle_1\rangle_2\langle\langle fh\rangle_1\langle ab\rangle_1\rangle_2\rangle_3$	$\langle(b,1)(c,1)(a,2)(c,1)(f,3)(h,1)(a,2)(b,1)\rangle$

Table 1.15: A multidimensional sequential database represented in both formats.

Customer id	Attribute $(A_{ME}, A_{MS}, A_P, A_{AG})$	Transaction
1	(4200,married,high-tech,24)	$\langle\{TM,CD\}\{WM\}\{RD,WM,TM\}\rangle$
2	(1500,single,retired,70)	$\langle\{CD,TM,WM\}\{CD,TM,WM\}\{WM\}\rangle$
3	(3800,married,teacher,32)	$\langle\{TM,CD\}\{CD,WM\}\{RD,TM\}\rangle$

Table 1.16: A sequence database representing activities of e-bank customers.

fixMDSpan for mining the simplified format. *AprioriMD* is an extension of *GSP* algorithm. The modification is done on two methods: (1) the method to generate C_k and (2) the method to compute the supports of candidate sequences. Another approach is *PrefixMDSpan* which extends *PrefixSpan* Pei et al. [2001] to deal with the multidimensional sequence data. The authors adapted the definitions of *prefix*, *projection*, *postfix*, and *projected database* to be suitable with the simplified format of sequential database. Then, the authors applied the usual steps of PrefixSpan taking into account the new definitions. The experiments showed that, the *PrefixMDSpan* algorithm outperforms the *AprioriMD* algorithm Yu and Chen [2005], which is similar to the relative performance trends between the *PrefixSpan* algorithm and the *GSP* algorithm as they did on one dimensional sequential database Pei et al. [2001]. The dimensions in Yu and Chen [2005] are very particular since they belong to a single hierarchical dimension. These three different dimensions can be projected into a single dimension corresponding to web pages, gathering web pages visited during a same session and ordering sessions w.r.t the day as order.

Stefanowski and Ziembinski [2005] and Ziembinski [2007] show the usefulness of multidimensional sequential patterns to study web usage mining. This study focuses on the behavior of users over e-bank, which helps the users to communicate with them by using Web portal and self service machine like WebATMs. In this system, the client executes several transaction like depositing money into bank account (CD), transferring money (TM), withdrawing money (WM), etc. The authors studied the sequential relation between customer transactions and took into account set of attributes describe each customer behaviors like monthly earning (A_{ME}), marital status (A_{MS}),etc. Table 1.16 shows an example of e-bank activities of customers. The authors showed the advantage of multidimensional sequential patterns to mine this kind of dataset. In addition, they suggest that the multidimensional attributes can be numeric or symbolic. However, no algorithm is defined in those two articles for mining a multidimensional sequential dataset.

MobilePrefixSpan algorithm proposed by Rashad et al. [2007] analyses information collected from the mobile users in cellular networks by generating frequent paths of mobile users. The authors used time intervals e.g., morning rush hours (T_1), morning-noon (T_2), after noon (T_3), etc and type of day, i.e., weekend or working day as the basic dimensions in this paper. The sequence represents transition of mobile user between cells. Table 1.17 shows an example of a database of multidimensional sequences collected by mobile user, for example, first sequence in this table represents transition of a mobile user from C_1 to C_3 and then C_2 at the first period of day (i.e, morning rush hours T_1) and during weekend (Y). First, *MobilePrefixSpan*

sid	dimension		sequence
	time interval	Weekend	
1	T_1	Y	$\langle\{C_1\}\{C_3\}\{C_2\}\rangle$
2	T_3	N	$\langle\{C_4\}\{C_1\}\{C_2\}\rangle$
3	T_1	Y	$\langle\{C_1\}\{C_4\}\{C_3\}\rangle$

Table 1.17: multidimensional sequences database collected by mobile user

Date D	Customer-Group CG	City C	Age A	Product P
1	Educ	NY	Y	c
1	Educ	NY	M	p
1	Educ	LA	Y	c
1	Ret	SF	O	c
1	Ret	SF	O	m
2	Educ	NY	M	p
2	Educ	NY	Y	r
2	Educ	LA	Y	c
3	Educ	LA	M	r

Table 1.18: Transactions issued by customers.

converts multidimensional sequential database to one-dimensional sequential database by adding two dimensions as an itemset of two items at the beginning of sequence. Then, *PrefixSpan* algorithm Pei et al. [2001] is applied. In *MobilePrefixSpan*, those paths are considered which contains only consecutively ordered cells required to move from one cell to another. Such kind of extracted sequences are called consecutive sequences. Actually, Rashad et al. [2007] uses the transformation of multidimensional sequence to one dimensional sequence which is defined at Pinto et al. [2001] and then applies one-dimensional sequential pattern mining method using *PrefixSpan* algorithm Pei et al. [2001] instead of proposing a new algorithm.

Plantevit et al. [2005] propose a novel approach for multidimensional sequential pattern mining called M^2SP (Mining Multidimensional Sequential patterns). It extracts patterns containing several dimensions over time, e.g., *"A customer who bought a surfboard and a bag in NY later bought a wetsuit in SF"*. This pattern not only combined two dimensions (City and Product) but it also combines them over time (NY appears befor SF, surfboard appears before wetsuit). Plantevit et al. [2005] illustrated the mechanism of M^2SP with an example transaction issued by customers. Here, the dataset has five dimensions which are:

1. The date of transactions, D.
2. The category of customers, CG.
3. The age of customers, A
4. The city where transactions have been issued, C.
5. The product of the transactions, P.

Table 1.18 shows an example to the dataset presented by Plantevit et al. [2005]. The first tuple means that, on date 1, educational young customers bought product c in New York. The

objective of M^2SP is extracting all multidimensional sequences where some dimensions are more frequent with respect to other dimensions. M^2SP aims at extracting all multidimensional sequences dealing with the age of customers, the products they bought, and that are frequent with respect the groups of customers and the cities where transactions have been issued. To solve this problem, M^2SP considers three dimensions: temporal dimension D_T, analysis dimension D_A, and reference dimension D_R. The support of the sequences is computed according to reference dimension. The tuples appearing in a sequence are defined over analysis dimension. The tuples in each block are ordered over temporal dimension.

For example, if we would like to extract all multidimensional sequences that deal with the age of customers and the products they bought. Thus, the analysis dimensions are age of customers dimension (A) and the product of the transactions dimension (P). And, the sequential patterns should be extracted with respect the groups of customers and the cities where transactions have been issued. Thus, the reference dimensions are the category of customers dimension (CG) and the city of transactions dimension (C).

A multidimensional sequence is defined as an ordered non empty set of multidimensional items where a multidimensional item is defined as:

Definition 27. *(Multidimensional Item)*
Let $D_A = \{D_1, D_2, ..., D_m\}$ be a set of m analysis dimensions. A multidimensional Item on D_A is a tuple $e = (d_1, d_2, ..., d_m)$ such that, for every $k \in [1, m]$, d_k is in the domain value of D_k.

In the first step, M^2SP splits the dataset into blocks according to distinct tuple values over reference dimensions. Figure 1.8 shows the three blocks obtained from Table 1.18 over two reference dimensions. Then, Plantevit et al. [2005] computes the support of a sequence by counting the number of blocks that support the sequence. In M^2SP, a block supports a sequence if all the multidimensional items in the sequence satisfy a set of tuples in the block and have the same order in both the sequence and the block. Plantevit et al. [2005] introduces a sequence containing at least one *jokerized item* called *"Jokerized Multidimensional Sequence"*. The *jokerized item* is a multidimensional item contains at least one specified analysis dimension and it contains the least number of *, i.e., it replaces any occurrence of * with any value from the corresponding domain cannot give a frequent sequence. This algorithm mines the frequent jokerized items by using a levelwise algorithm Beyer and Ramakrishnan [1999] to build the frequent multidimensional items having the smallest possible number of *. Figure 1.9 illustrates an example of mining frequent jokerized items by using a level wise algorithm and the Blockes in Figure 1.8 with minimum support threshold $\frac{2}{3}$ (i.e., the item is frequent if two blocks of the all the three blocks support it). M^2SP only focuses on the most specific multidimensional items instead of mining at all possible items. M^2SP associated each multidimensional item extracted with a unique identifier, i.e., every block b is assigned an identifier then every tuple in the block is replaced with its identifier. Finally, PSP Masseglia et al. [1998] is applied to mine sequential pattern from transformed sequential dataset.

1.5.2 Multi Level Sequential Patterns Methods

In real world applications, many datasets have a user-defined taxonomy over the items in the data, An example of a taxonomy is given in Figure 1.10 which is applied to market basket analysis. For instance, we can see that neither apple nor orange is frequent in the sequential database whereas the value fruit is frequent. The interesting problem here is to extract more or less general/specific sequential patterns and overcome the limitation of excessive granularity and low support.

Date	Age	Product
1	Y	c
1	M	p
2	M	p
2	Y	r

a. Block (Educ,NY)

Date	Age	Product
1	Y	c
2	Y	c
3	M	r

b. Block (Educ,LA)

Date	Age	Product
1	O	c
1	O	m

c. Block (Ret,SF)

Figure 1.8: Blocks defined on Table 1.18 over two reference dimensions.

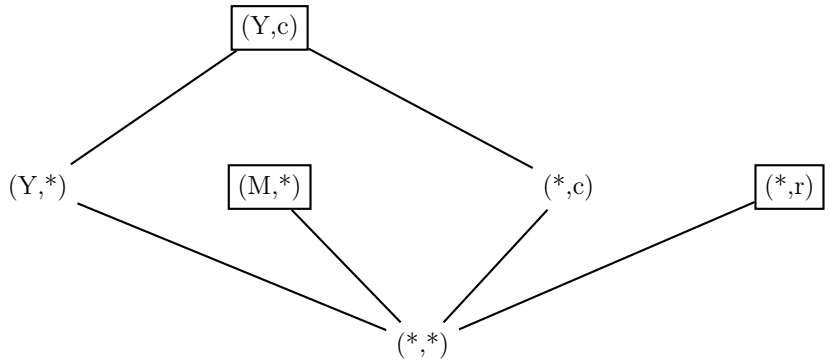


Figure 1.9: The frequent jokerized items

Pinto et al. [2001] used taxonomy in *GSP* to mine more sequential patterns. The proposed approach is based on extending the sequential database by using the taxonomy as follows. Replace every item in the sequence with all the ancestors in the taxonomy, then apply *GSP* to mine sequential patterns from the extended sequential database. For instance, the sequence $\langle\{apple, white\ win\}\{orange\}\rangle$ can be extended by using the taxonomy in Figure 1.10 to the sequence $\langle\{product, food, fresh\ food, fruit, apple, drink, alcohol\ drink, win, white\ win\}$ $\{product, food, fresh\ food, fruit, orange\}\rangle$. This solution is not scalable because the size of the database becomes the product of maximum height of the taxonomy. To improve the performance, this algorithm proposes two optimizations. First is to compute the ancestors of each item a priori and drop ancestors which are not in any of the candidates being counted before making a pass over the data. The second way is to ignore sequential patterns having an element with both an item and its ancestor as when the sequence with the item is frequent the sequence with its ancestors will be also frequent.

Han and Fu [1999] proposes a method for mining multiple level association rules in large databases, so that their approach can be adapted for mining multiple level sequential patterns. This work is different because it deals with association rules but it is applicable for multilevel sequential patterns. The proposed approach starts at the highest level of the taxonomy to which the items belong. The rules on each level are extracted while lowering the support when going down in the taxonomy. The process is repeated until no rule can be extracted any more or until the lowest level of the taxonomy is reached. However, this method does not allow the extraction of sequential patterns containing items from different levels of the taxonomy. For example apple and fruit cannot appear together in a sequential pattern. Thus, this approach does not address the general problem of extracting sequences including elements at different levels of granularity.

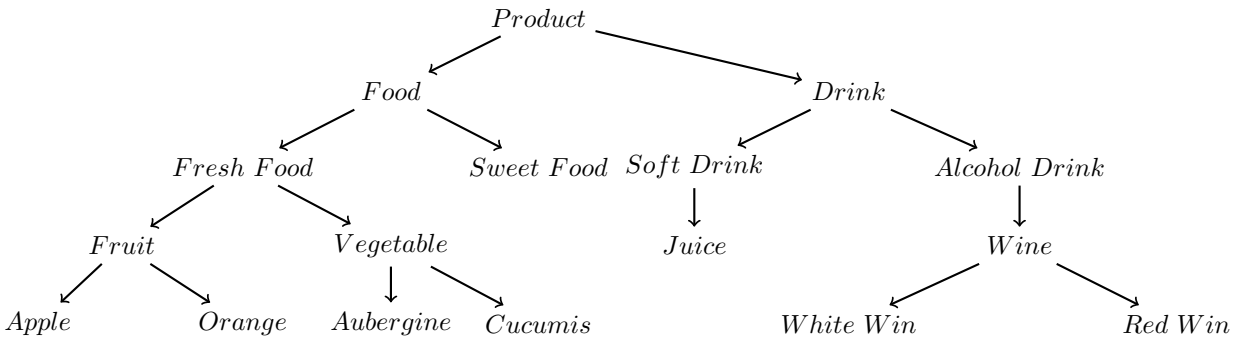


Figure 1.10: Hierarchy over product.



Figure 1.11: The difference between M^2SP Plantevit et al. [2005] and M^3SP Plantevit et al. [2010] in handling the taxonomy.

1.5.3 Multi Dimensional and Multi Level Sequential Patterns Methods

Plantevit et al. [2010] proposes a method for mining sequential patterns from multidimensional databases called , M^3SP. It simultaneously takes advantage of the different levels of granularity for each dimension (taxonomy). The most general item in the taxonomy is denoted by ALL. M^3SP is an extension of M^2SP Plantevit et al. [2005]. The difference between M^2SP and M^3SP is that the former goes directly from the finest level in the taxonomy to the most general level ALL (which is denoted as * in Plantevit et al. [2005]) while the later goes to the most general level by passing the taxonomy level by level (see Figure 1.11) .

M^3SP uses all the definitions of M^2SP and the same concept of dividing the multidimensional sequential database into blocks but it introduces the notion of the maf sequence.

Definition 28. *(maf sequences)*

The atomic sequence $\langle\{a_1\}\rangle$ is said to be a maximal atomic sequence, or a maf-sequence, if $\langle\{a_1\}\rangle$ is frequent and if for every multidimesional item a_2 such that a_2 is more specific than a_1, the sequence $\langle\{a_2\}\rangle$ is not frequent.

M^3SP is composed of two steps:

- Maf-sequences are mined by using a strategy inspired from the BUC algorithm Beyer and Ramakrishnan [1999].
- All frequent sequences that can be built up based on the maf-sequences found in the previous step are mined by using SPADE algorithm Zaki [2001].

A maf-sequence is generated by firstly computing all the sequences of the form $\langle\{(ALL_1, ..., ALL_{i-1}, d_i, ALL_{i+1}, ..., ALL_m\}\rangle$, where ALL_i is the most general item in the ith analysis dimension's taxonomy, m is the number of analysis dimension and d_i is the direct descendant of the ALL in the ith analysis dimension. Then, M^3SP prunes from the taxonomy of each analysis dimension all values d_i and their descendants such that $\langle\{(ALL_1, ..., ALL_{i-1}, d_i, ALL_{i+1}, ..., ALL_m\}\rangle$ is not frequent. Based on the frequent maf sequence extracted in previous step, M^3SP uses a recursive function called *get_rec_maf_seq* to generate all the frequent maf-sequences (see Algorithm 1). Afterwards, frequent sequences are mined with the help of SPADE algorithm Zaki [2001] which first assigns an identification to all the frequent maf-sequences and then recodes the multidimensional database by using the identification assigned to the frequent maf-sequence.

Algorithm 1: Routine *get_rec_maf_seq*

input : A multidimensional item a, the minimum support threshold minsup, the set of blocks $B(D_R)$.

```
1 begin
2   Cand ← {a = (d_1, ..., d_m) ∈ gen(a)|(∀i = 1, ..., m)(count(d_i) ⩾ minsup)} ;
3   Freq ← {a ∈ Cand |sup(a) ⩾ minsup)} ;
4   if Freq ≠ ∅ then
5     MAFS ← MAFS ∪{⟨{a}⟩} ;
6   else
7     foreach a ∈ Freq do
8       call get_rec_maf_seq(a,minsup,σ_a(B(D_R)));
```

1.6 Discussion

Table 1.19 gives a brief overview of all the sequential pattern extraction approaches previously presented. Sequential patterns have been proposed to extract correlations between events following an order relation (i.e., time) Srikant and Agrawal [1996]. In this work, correlation between values of the same dimension over time is highlighted. There are several propositions for the extraction of frequent sequences of one dimension like Srikant and Agrawal [1996], Masseglia et al. [1998], Zaki [2001], Pei et al. [2001] and Yin and Han [2003], etc. To our knowledge, Srikant and Agrawal [1996] is the only study that takes into account the sequences of one dimension defined on several levels of hierarchies.

Sequential patterns do not take into account the multidimensionality. That is why the multidimensional sequential patterns were defined in Pinto et al. [2001]. This approach extracts frequent sequences defined on one dimension and characterize the frequent sequence by a multidimensional pattern. The sequences found by this approach do not contain several dimensions, since, the time is relative to a single dimension. In addition, this proposal and those arising fail to take into account the hierarchies.

The approach of Stefanowski and Ziembinski [2005] takes into account numerical attributes from the definition of Pinto et al. [2001] by introducing similarity measures to calculate the support of a sequence. However, this proposal is not supported by any definition or algorithm.

The approach of Yu and Chen [2005] extracts sequences in multidimensional sequential dataset. This study considers pages, sessions and days as dimensions. Actually, these three

		Sequential Patten Mining				
		Approach		item	itemset	multilevel
One-dimensional	Horizontal	GSP	Srikant and Agrawal [1996]		√	√
		PSP	Masseglia et al. [1998]		√	
		MFS	Zhang et al. [2001]		√	
		MSPX	Luo and Chung [2008]		√	
	Vertical	SPADE	Zaki [2001]		√	
		cSPADE	Zaki [2000a]		√	
		SPAM	Ayres et al. [2002]		√	
		CCSM	Orlando et al. [2004]		√	
		LAPIN-SPAM	Yang and Kitsuregawa [2005]		√	
	Projection	FreeSpan	Han et al. [2000a]		√	
		PrefixSpan	Pei et al. [2001]		√	
	Closed	CloSpan	Yin and Han [2003]		√	
		BIDE	Wang and Han [2004]		√	
		ClaSP	Gomariz et al. [2013]		√	
	Consensus	ApproxMAP	Kum et al. [2003]		√	
		ProMFS	Tumasonis and Dzemyda [2004]		√	
	Parallel	NPSPM SPSPM HPSPM	Guralnik and Karypis [2004]		√	
		webSPADE	Demiriz [2002]		√	
		Par-CSP	Cong et al. [2005]		√	
Multi-dimensional		UniSeq Dim-Seq Seq-Dim	Pinto et al. [2001]	√		
		AprioriMD PrefixMDSpan	Yu and Chen [2005]		√	
		M^2SP	Plantevit et al. [2005]	√		
		M^3SP	Plantevit et al. [2010]	√		√

Table 1.19: Sequential pattern mining methods.

different dimensions can be projected into a single dimension corresponding to web pages, web pages visited gathering falling on a same session and ordering sessions w.r.t the day as order.

The approaches of Plantevit et al. [2005] extracts sequential pattern in a multidimensional sequential dataset. These approaches take both aspects into account where each dimension is represented as an item at different levels of granularity, by using a taxonomy. In M^3SP, Plantevit et al. [2005] represents the multidimensional event as a tuple of atomic items. This restriction prevents M^3SP from extracting condensed and heterogeneous patterns.

In modern life sciences, a multidimensional sequential data set is often represented as sequences of vectors with elements having different types (i.e., *item* and *itemset*). This special feature is in itself a challenge and multidimensionality in sequence mining needs to be carefully taken into account when devising new efficient algorithms. To our knowledge, there is no proposal aiming at the extraction of rules which are ordered over time and are represented with different levels of granularity, and it also characterizes the dimensions in a different way.

Chapter 2

Mining Heterogeneous Multidimensional Sequential Patterns

Contents

2.1 Introduction 35
2.2 Problem Statement 36
2.2.1 An introductory example 36
2.2.2 Basic definitions 37
2.2.3 Most specific multidimensional sequential patterns 39
2.3 MMISP algorithm 40
2.3.1 Step 1: Extracting all frequent elementary vectors: 40
2.3.2 Step 2: Transformation of mds-database: 45
2.3.3 Step 3: mds-patterns mining: 47
2.4 Implementation and Experimental Validation 47
2.4.1 Experiments on Synthetic Datasets 48
2.5 Conclusion and perspective 51

2.1 Introduction

Sequential pattern mining, introduced by Agrawal et al Agrawal and Srikant [1995], is a popular approach to discover patterns in ordered data. Frequent sequence mining can be seen as an extension of the well known itemset mining problem where the input data is modeled as sequences of elements. This method is rather efficient to discover rules of the type: *"customers frequently first buy DVDs of seasons I, II of Sherlock, then buy within 6 months season III of the same crime drama series"*. Sequential pattern mining has been successfully used so far in various domains : amino-acids protein sequence analysis Chothia and Gerstein [1997], web log analysis Yang and Zhang [2003], and music sequences matching Serrà et al. [2012].

As we see in Chapter 1, many efficient approaches were developed to mine patterns depending on time or order where most of them are based on the *Apriori* property Agrawal and Srikant [1995], which states that any super pattern of a non-frequent pattern cannot be frequent. The main algorithms are *GSP* Srikant and Agrawal [1996], *SPADE* Zaki [2001], *PrefixSpan* Pei et al. [2004b] *ClosSpan* Yan et al. [2003]. However, these techniques and algorithms focus solely on

one-dimensional aspect of sequential databases and do not deal with the multidimensional aspect where items can be of different types and described over different levels of granularity. For instance, in a real-world retail company, a database or data warehouse holds much more complex information such as article prices, gender of the customers, geolocation of the stores and so on. In addition, articles are usually represented following a hierarchical taxonomy: apples can be either described as fruits, fresh food or food. Pinto et al. [2001], Zhang et al. [2007] and Yu and Chen [2005] introduced the notion of multidimensionality in a sequence and proposed several algorithms to mine this type of data without taking into account the different levels of granularity for each dimension. Plantevit et al. [2005] introduced M^3SP, an algorithm able to incorporate several dimensions described over different levels of granularity within the sequential pattern mining process. These approaches focus on homogeneous multidimensional sequence where its elements are described simply as vectors of items. By contrast, in modern life sciences Wodak and Janin [2002], a multidimensional sequential data set is often represented as sequences of vectors with elements having different types (i.e., *item* and *itemset*). This special feature is in itself a challenge and multidimensionality in sequence mining needs to be carefully taken into account when devising new efficient algorithms.

In our approach, we aim at providing an approach that extract rules such as: *"After buying an article from the fruit category from supermarket A, a customer will buy two articles from the Egg and Dairy products and Beverage categories from supermarket B"*. This example not only combines two dimensions (supermarket and products) which are ordered over time and are represented with different levels of granularity, but it also characterizes them in a different way as a magazine can be considered as an element taking one value (*"item"*), while a product can take several values (*"itemset"*). This example shows that each dimension has to be managed in a proper and suitable way. We believe that our work is the first to present a full framework and algorithm to mine such multidimensional sequential patterns from heterogeneous multidimensional sequential database.

The main contribution of this chapter is to generalize the concept of multidimensional sequence by considering heterogeneous multidimensional sequences. The event in a sequence is considered as a vector of items and itemsets, Such multidimensional and heterogeneous patterns have to be mined by adapted and suitable methods. Accordingly, we propose a new method *MMISP (Mining Multidimensional Itemsets Sequential Patterns)* to extract sequential patterns from heterogeneous multidimensional sequential database. In addition, the approach is able to take into account background knowledge lying in taxonomies existing for each dimension. As often with enumeration algorithms, mining all possible sequential patterns from a multidimensional sequential database results in a huge amount of patterns which is difficult to be analyzed Raïssi and Pei [2011]. To overcome this problem, *MMISP* mines only the most specific patterns. We report an extensive empirical study on synthetic datasets

The remainder of this chapter is organized as follow, Section 2.2 introduces the problem statement and a running example. The algorithm for extracting sequential patterns is described in Section 2.3. Section 2.4 presents experimental results from quantitative points of views. Section 2.5 concludes this approach.

2.2 Problem Statement

2.2.1 An introductory example

In this section, we propose an example to illustrate and ease the understanding of our proposed approach. The example focus on mining patient trajectory in a healthcare system. A patient

Patients	Trajectories
s_1	$\langle(uh_p, ca_1, \{mp_{111}, mp_{221}\}), (uh_p, ca_1, \{mp_{222}\}), (gh_l, r_1, \{mp_{221}, mp_{311}\})\rangle$
s_2	$\langle(uh_n, ca_1, \{mp_{111}\}), (uh_n, ca_2, \{mp_{111}, mp_{211}\}), (gh_l, r_1, \{mp_{222}, mp_{312}\})\rangle$
s_3	$\langle(uh_n, ca_3, \{mp_{112}, mp_{211}\}), (gh_l, r_2, \{mp_{222}, mp_{311}\})\rangle$
s_4	$\langle(uh_p, ca_2, \{mp_{112}, mp_{222}\}), (gh_p, r_2, \{mp_{221}, mp_{312}\}), (gh_p, r_2, \{mp_{221}, mp_{312}\})\rangle$

Table 2.1: An example of a database of patient trajectories.

trajectory can be considered as a sequence of hospitalizations ordered over time, where each time stamp corresponds to one hospitalization. This hospitalization is represented as a vector of 3 elements. Each vector represents specific information about one stay of a patient in a hospital:

- The hospital where a patient is admitted.
- A reason for hospitalization.
- Set of medical procedures that a patient undergoe

Table 2.1 describes four patient trajectories. For example, s_1 is a patient trajectory with three hospitalizations and the vector $(uh_p, ca_1, \{mp_{111}, mp_{221}\})$ fully describes the first hospitalization of patient s_1 who was admitted to the hospital uh_p for treatment of lung cancer ca_1, and underwent procedures mp_{111} and mp_{221}. In a patient trajectory, background knowledge is usually available in form of taxonomies, classification or concept hierarchies. Each element in the hospitalization can be represented at different levels of granularity, by using a taxonomy (see Figure 2.1).

Our goal is to find specific patterns that appears frequently in patients trajectories', by taking advantage of different levels of granularity for each element as background knowledge. Such patterns are very helpful to improve hospitalization planning, optimize clinical processes or detect anomalies.

2.2.2 Basic definitions

We assume in the following that domain knowledge always exists in the form of taxonomies. A sequence is constituted of an alphabet which is a vector of elements. This element is either an item or a set of items, where each item is a node in a taxonomy. For the sake of simplicity we call a multidimensional sequence a *"md-sequence"* and we define it as follows:

Definition 29. *(md-sequence)*
A md-sequence $s{=}\langle s_1, s_2, ..., s_n\rangle$ is defined as set of elementary vectors $s_i = (e_1, ..., e_k)$ ordered by the temporal order relation $<_t$ such as $s_1 <_t s_2 <_t s_3 ... <_t s_n$, where n is called the size of the sequence s; i.e., $|s| = n$.

The vector $e = (e_1, ..., e_k)$ is ***more specific*** than $e' = (e'_1, ..., e'_k)$, denoted by $e \leqslant e'$, iff $e_i \leqslant e'_i$ for all $i \leqslant k$. Then, ordering over md-sequence is defined as follows

Definition 30. *(Ordering of Multidimensional Sequences)*
*Given two md-sequences $s = \langle s_1, s_2, ..., s_{n_1}\rangle$ and $s' = \langle s'_1, s'_2, ..., s'_{n_2}\rangle$, the sequence $s = \langle s_1, s_2, ..., s_{n_1}\rangle$ is **more specific** than $s' = \langle s'_1, s'_2, ..., s'_{n_2}\rangle$, denoted by $s \leqslant s'$, if there exist a set of indices $1 \leqslant i_1 < i_2 < ... < i_{n_2} \leqslant n_1$ such that $s_j \leqslant s'_{i_j}$ for all $j \in \{1 \dots n_2\}$ and $n_2 \leqslant n_1$. s' is said to be **more general** than s.*

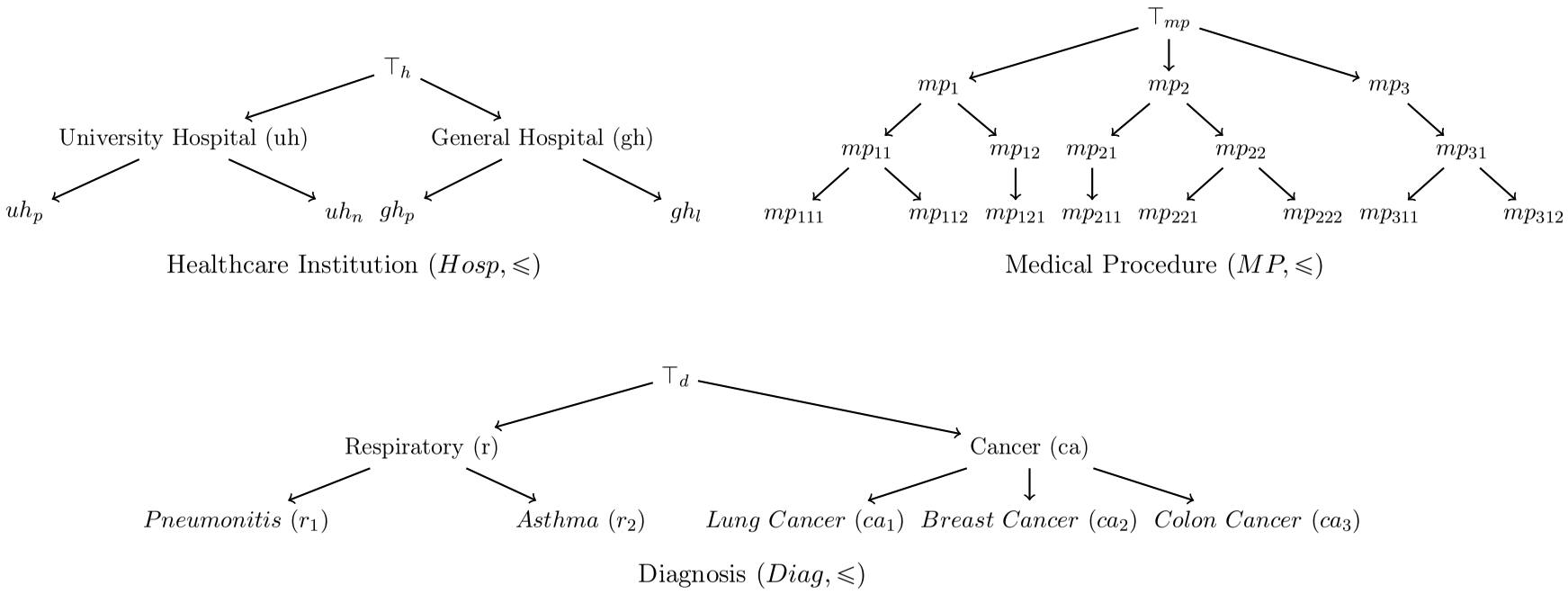


Figure 2.1: Taxonomies for the healthcare institution, the medical procedure and the diagnosis

Example 11. *Consider the three taxonomies* $(Hosp, \leqslant)$, $(Diag, \leqslant)$ *and* $(MP, \leqslant)$ *in Figure 2.1. The elementary vector* $e = (uh_p, ca_1, \{mp_{111}, mp_{121}\})$ *is more specific than* $e' = (uh, ca, \{mp_{11}, mp_{12}\})$, *as* $uh_p \leqslant uh$, $ca_1 \leqslant ca$ *and* $\{mp_{111}, mp_{121}\} \leqslant \{mp_{11}, mp_{12}\}$. *The sequence* $s = \langle (uh_p, ca_1, \{mp_{111}, mp_{121}\}), (gh_l, r_2, \{mp_{121}, mp_{131}\})\rangle$ *is a md-sequence with two elementary vectors* $s_1 = (uh_p, ca_1, \{mp_{111}, mp_{121}\})$ *and* $s_2 = (gh_l, r_2, \{mp_{121}, mp_{131}\})$ *where* s_1 *comes before* s_2 *over time. The sequence* $s' = \langle (uh, ca_1, \{mp_{11}, mp_{12}\}), (gh_l, r, \{mp_{13}\})\rangle$ *is more general than* s, *as* $s_1 \leqslant s'_1$, $s_2 \leqslant s'_2$ *and* s_1 *and* s'_1 *come before* s_2 *and* s'_2 *over time.*

Given a set $\mathcal{MS}_{\mathcal{DB}} = \{s_1, s_2, ..., s_m\}$ of m md-sequences. The set $\mathcal{MS}_{\mathcal{DB}}$ is called a *mds-database*. The support of an elementary vector in a mds-database is defined as follows:

Definition 31. *(Support of an elementary vector)*
Let $\mathcal{MS}_{\mathcal{DB}}$ *be a mds-database and let* $e = (e_1, e_2, ..., e_k)$ *be an elementary vector. The support of the elementary vector* e *in* $\mathcal{MS}_{\mathcal{DB}}$, *denoted by* $supp(e, \mathcal{MS}_{\mathcal{DB}})$, *is defined as follows:*

$$supp(e, \mathcal{MS}_{\mathcal{DB}}) = |\{s_i \in \mathcal{MS}_{\mathcal{DB}};\ \exists j \leqslant |s_i|;\ s_{ij} \leqslant e\}|$$

While the support of a sequence s in $\mathcal{MS}_{\mathcal{DB}}$ is defined as follows:

Definition 32. *(Support of md-sequence)*
Let $\mathcal{MS}_{\mathcal{DB}}$ *be a mds-database and let* s *be a md-sequence. The support of* s, *denoted by* $supp(s, \mathcal{MS}_{\mathcal{DB}})$ *is defined as follows:*

$$supp(s, \mathcal{MS}_{\mathcal{DB}}) = |\{s_i \in \mathcal{MS}_{\mathcal{DB}}; s_i \leqslant s\}|$$

Given a positive integer σ as minimal support threshold, and a mds-database $\mathcal{MS}_{\mathcal{DB}}$, the elementary vector e is called frequent, iff $supp(e, \mathcal{MS}_{\mathcal{DB}}) \geqslant \sigma$. A md-sequence is *frequent* in $\mathcal{MS}_{\mathcal{DB}}$ if its support in $\mathcal{MS}_{\mathcal{DB}}$ exceeds the minimal support threshold σ. A frequent md-sequence is called a *"mds-pattern"*. The problem of mining md-sequences is to enumerate all possible mds-patterns, given a mds-database $\mathcal{MS}_{\mathcal{DB}}$ and a minimal support threshold.

Example 12. *The sequence* $s = \langle (uh, ca, \{mp_{11}, mp_2\}), (\top_h, \top_d, \{mp_{222}\})\rangle$ *has a support which is equal to 3 (i.e.,* $supp(s, \mathcal{MS}_{\mathcal{DB}}) = 3$*) in the database* $\mathcal{MS}_{\mathcal{DB}}$ *(see Table 2.1) and is a mds-pattern w.r.t to the minimal support (i.e.,* $\sigma = 3$*).*

2.2.3 Most specific multidimensional sequential patterns

In this section, we present the problem of mining *the most specific mds-patterns*. Mining all possible mds-patterns from $\mathcal{MS}_{\mathcal{DB}}$ results in a huge amount of patterns that is difficult to manage. Thus, we extract a set of mds-patterns that are not only frequent but also the most specific. This second constraint allows the reduction of the number of returned sequences by discarding patterns that are *"too general"*. The most specific mds-pattern is defined as follows:

Definition 33. *(Most specific mds-pattern)*
Given a positive integer σ *as minimal support threshold and a mds-database* $\mathcal{MS}_{\mathcal{DB}}$. *The sequence* s *is a most specific mds-pattern in* $\mathcal{MS}_{\mathcal{DB}}$ *if and only if* $supp(s, \mathcal{MS}_{\mathcal{DB}}) \geqslant \sigma$ *and there does not exist any sequence* s' *such that* $supp(s', \mathcal{MS}_{\mathcal{DB}}) \geqslant \sigma$ *and* $s' \leqslant s$.

In this precise setting, the frequency for sequences is *monotone*; i.e., whenever s is frequent, any generalization of s is also frequent. For example, if $s = \langle (uh_p, ca_1, \{mp_{111}, mp_{112}\})\rangle$ is frequent in $\mathcal{MS}_{\mathcal{DB}}$ then $s' = \langle (uh, ca, \{mp_1\})\rangle$ which is more general than s is also frequent.

Thus, the most specific sequential patterns constitute a basic for retrieving all sequential patterns. Let $\sigma = 3$ be a minimal support threshold, the sequence $s = \langle(uh, \top_d, \{mp_1\}), (\top_h, \top_d, \{mp_2\})\rangle$ is frequent, but is not the most specific one as the sequence $s' = \langle(uh, ca, \{mp_{11}, mp_2\}), (gh, r, \{mp_{22}, mp_{31}\})\rangle$ is frequent and verifies that $s' \leqslant s$. The sequence s' is a most specific mds-pattern as it is frequent and there is no other sequence in $\mathcal{MS}_{\mathcal{DB}}$ which is frequent and more specific.

The problem of mining mds-patterns is reduced to discover only the most specific patterns to significantly decrease the complexity of the problem and save computational time.

2.3 MMISP algorithm

In this section, we present an approach for extracting the most specific mds-patterns from a mds-database $\mathcal{MS}_{\mathcal{DB}}$. Our approach is called ***MMISP (Mining Multidimensional Itemsets Sequential Patterns)***. The basic idea of *MMISP* consists in transforming the mds-data into a *"classical form"* (i.e., sequence of itemsets) and then applying a standard algorithm for sequential pattern mining. *MMISP* is based on three steps:

1. ***Extraction of frequent elementary vectors***
   The algorithm searches for the frequent and specific elementary vectors.

2. ***Transformation***
   In this step, all frequent elementary vectors are mapped into an alternate representation, then, the mds-database is encoded by using this new representation.

3. ***mds-patterns mining***
   In this step, a standard sequential algorithm is applied to the sequential database produced at the preceding step.

2.3.1 Step 1: Extracting all frequent elementary vectors:

The basic step in *MMISP* is extracting all frequent elementary vectors from $\mathcal{MS}_{\mathcal{DB}}$ w.r.t. the ordering relation existing between their elements.

If the elementary vector is infrequent, then neither it nor its specifications will appear in mds-patterns. Thus, *MMISP* extracts only the frequent elementary vectors from $\mathcal{MS}_{\mathcal{DB}}$ to find all the most specific mds-patterns. The main challenge in this step is *how to efficiently mine the frequent elementary vectors.*

Assume that we have an elementary vector, composed of k bottom elements (i.e., $\perp = (\perp_1, ..., \perp_k)$), is more specific than any other elementary vector. If element of the vector is an item, then we consider a node $\perp_i$ connected by edges to all leaves of taxonomy as a bottom node. Otherwise if the element is an itemset, then the bottom is a set of all the leaf nodes. In our running example, the bottom elementary vector is $(\perp_h, \perp_d, \{mp_{111}, mp_{112}, mp_{121}, mp_{211}, mp_{221}, mp_{222}, mp_{311}, mp_{312}\})$. The set of all the elementary vectors E with the bottom vector $\perp$ is a *lattice* $(E \cup \{\perp\}, \leqslant)$. Given two elementary vectors $e = (e_1, ..., e_k)$ and $e' = (e'_1, ..., e'_k)$ in E, the join $(\sqcup)$ of e and e' is defined as the join of ith element in e and e'; i.e., $e \sqcup e' = (e_1 \sqcup e'_1, ..., e_k \sqcup e'_k)$. The join of two nodes in a taxonomy is the lowest common ancestor of these nodes, while the join of two sets of nodes $c = \{c_1, ..., c_n\}$ and $c' = \{c'_1, ..., c'_m\}$ is the most specific values from the set $\{\forall(i, j);\ c_i \sqcup c'_j\}$; $i \leqslant n$ and $j \leqslant m$.

Example 13. *Consider the three taxonomies* $(Hosp, \leqslant)$, $(Diag, \leqslant)$ *and* $(MP, \leqslant)$ *in Figure 2.1, the two elementary vectors* $e = (uh_p, ca_2, \{mp_{111}, mp_2\})$ *and* $e' = (uh_n, ca, \{mp_{121}, mp_{21}\})$. *The*

join of e and e′, $e \sqcup e'$, is $(uh_p \sqcup uh_n, ca_2 \sqcup ca, \{mp_{111}, mp_2\} \sqcup \{mp_{121}, mp_{21}\})$, where the join of uh_p and uh_n is uh , ca_2 and ca is ca and the join of $\{mp_{111}, mp_2\}$ and $\{mp_{121}, mp_{21}\}$ is the set $\{mp_1, mp_2\}$. Thus, the join of e and e′ is the elementary vector $(uh, ca, \{mp_1, mp_2\})$.

The meet ($\sqcap$) of two elementary vectors e and e' is $e \sqcap e' = (e_1 \sqcap e'_1, ..., e_k \sqcap e'_k)$. The meet between two nodes in a taxonomy is the most specific one if they are comparable, otherwise it is the bottom node $\perp_i$. The meet of two sets of nodes $c = \{c_1, ..., c_n\}$ and $c' = \{c'_1, ..., c'_m\}$ is the most specific values from the set of the ancestors of each item in the two sets c and c'.

Example 14. *Consider the three taxonomies $(Hosp, \leqslant)$, $(Diag, \leqslant)$ and $(MP, \leqslant)$ in Figure 2.1, the two elementary vectors $e = (uh_p, ca_2, \{mp_{111}, mp_2\})$ and $e' = (uh_n, ca, \{mp_{121}, mp_{21}\})$. The meet of e and e′, $e \sqcap e'$, is $(uh_p \sqcap uh_n, ca_2 \sqcap ca, \{mp_{111}, mp_2\} \sqcap \{mp_{121}, mp_{21}\})$, where the meet of uh_p and uh_n is $\perp_h$, ca_2 and ca is ca_2 and the meet of $\{mp_{111}, mp_2\}$ and $\{mp_{121}, mp_{21}\}$ is the set $\{mp_{111}, mp_{121}, mp_{21}\}$. Thus, the meet of e and e′ is the elementary vector $(\perp_h, ca_2, \{mp_{111}, mp_{121}, mp_{21}\})$.*

Finally, we can say that $(E \cup \{\perp\}, \leqslant)$ is a *lattice*, while the frequent elementary vectors considers as a *join-semilattice* $(FE, \leqslant)$. The main challenge now is how to efficiently build the join-semilattice $(FE, \leqslant)$. This task is achieved through a *depth-first*, and from *left-to-right* traversal Zaki et al. [1997], starting from the most general elementary vector $\top = (\top_1, ..., \top_k)$, we consider it as frequent elementary vector. Then, for each frequent elementary vector e, we recursively generate all the immediate successors of e, and for each of them, we compute its support in $\mathcal{MS}_{\mathcal{DB}}$ and we keep the frequent one.

We need to define an effective and nonredundant way to characterize the immediate successors in $(FE, \leqslant)$ of a given elementary vector $e = (e_1, ..., e_k)$ as follows. Firstly, we will assume that elements of the elementary vector are ordered according to a fixed total ordering. We will define an index z over the elementary vector to generate its immediate successors without redundancy and to build $(FE, \leqslant)$ from left to right. This index is defined as the position of the element in e which is more specific than $\top_i$ and all the elements after this one until end of e are $\top_i$ (in the case of e is $(\top_1, ..., \top_k)$, the index z equals to 1).

Example 15. *Given the elementary vector $e = (\top_h, ca, \top_{mp})$, then the index z equals to 2 as the second element is not $\top_d$; i.e., $e_2 = ca$, and all the elements after ca until end of e are top; i.e., $e_3 = \top_{mp}$.*

To generate the immediate successors of an elementary vector e, we substitute each element in e, which has its position greater than or equal to the index z, with one of its immediate successors and the rest of the elements are kept as it is.

Example 16. *Given the same previous example $e = (\top_h, ca, \top_{mp})$, as we see that the index z in e is equal to 2, then its immediate successors are consisted of two sets. The first one contains all the elementary vectors which are generated by substituting the second element ca with one of its immediate successors and keeping the first and the third element; i.e., $\top_h$ and $\top_{mp}$ respectively. While the second set contains all the elementary vectors which are generated by substituting the third element $\top_{mp}$ with one of its immediate successors and keeping the first and the second element; i.e., $\top_h$ and ca respectively.*

The immediate successors of an element depend on its type, if an element is an item then we follow the standard definition of immediate successor in the taxonomy. For example, given the taxonomy $(Diag, \leqslant)$ in Figure 2.1, the immediate successors of *ca* are ca_1, ca_2 and ca_3.

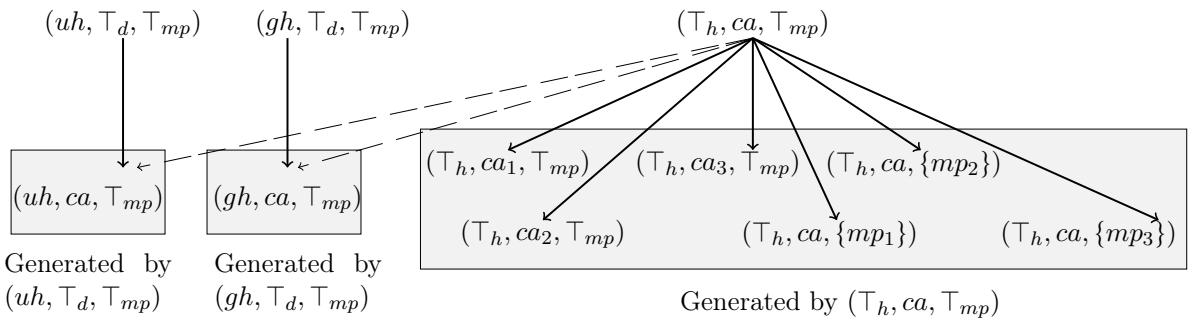


Figure 2.2: The immediate successors of $(\top_h, ca, \top_{mp})$

In case, the element is an itemset $c = \{c_1, c_2, ..., c_m\}$, then to generate nonredundant immediate successors of c, we assume that its items are ordered according to a fixed total ordering. The immediate successors of c are splitted into two sets. The first one is generated by substituting the last item in c; i.e., c_m, with one of its immediate successors and the rest of the items are kept as it is; i.e., the first set of the immediate successors of c contains an itemset $c' = \{c'_1, ..., c'_m\}$ where $c'_i = c_i$ for all $i < m$ and c'_m is one of the immediate successors of c_m. The second set is generated by adding new item c_{m+1} to end of c, where c_{m+1} is one of the right siblings of c_m and all its ancestor nodes; i.e., the second set of the immediate successors of c contains an itemset $c' = \{c'_1, ..., c'_m, c'_{m+1}\}$ where $c'_i = c_i$ for all $i \leqslant m$ and c'_{m+1} is one of the right siblings of c_m and all its ancestor nodes.

Example 17. *Given the taxonomy $(MP, \leqslant)$ in Figure 2.1, the immediate successors of $c = \{mp_1, mp_{21}\}$ are generated by substituting the last item mp_{21} in c with its immediate successors; i.e., $\{mp_1, mp_{211}\}$, and by adding the right siblings of mp_{21} and all its ancestor nodes; i.e., mp_{22} and mp_3, to the end of c; i.e., $\{mp_1, mp_{21}, mp_{22}\}$ and $\{mp_1, mp_{21}, mp_3\}$.*

Given the taxonomies in Figure 2.1, the immediate successors of $e = (\top_h, ca, \top_{mp})$ are $(uh, ca, \top_{mp})$, $(gh, ca, \top_{mp})$, $(\top_h, ca_1, \top_{mp})$, $(\top_h, ca_2, \top_{mp})$, $(\top_h, ca_3, \top_{mp})$, $(\top_h, ca, \{mp_1\})$, $(\top_h, ca, \{mp_2\})$ and $(\top_h, ca, \{mp_3\})$. The first two are generated from $(uh, ca, \top_{mp})$ and $(gh, ca, \top_{mp})$ respectively, while the rest are generated from $(\top_h, ca, \top_{mp})$. First by replacing only ca with one of ca_1, ca_2 and ca_3 and keeping $\top_h$ and $\top_{mp}$; i.e., $(\top_h, ca_1, \top_{mp})$, $(\top_h, ca_2, \top_{mp})$ and $(\top_h, ca_3, \top_{mp})$. Then, by replacing only $\top_{mp}$ with one of $\{mp_1\}$, $\{mp_2\}$ and $\{mp_3\}$ and keeping $\top_h$ and ca; i.e., $(\top_h, ca, \{mp_1\})$, $(\top_h, ca, \{mp_2\})$ and $(\top_h, ca, \{mp_3\})$ (see Figure 2.2).

The frequency of an elementary vector is monotone, the specialization of a non-frequent elementary vector is also non-frequent. We use this monotonicity to prune the enumeration space and efficiently build the semilattice $(FE, \leqslant)$. Figure 2.3 shows an example of generation of a part of $(FE, \leqslant)$ with $\sigma = 3$ which is detailed as follows. As the first step we consider the most general elementary vector $(\top_h, \top_d, \top_{mp})$, from which seven new frequent elementary vectors, $(uh, \top_d, \top_{mp})$, $(gh, \top_d, \top_{mp})$, $(\top_h, r, \top_{mp})$, $(\top_h, ca, \top_{mp})$, $(\top_h, \top_d, \{mp_1\})$, $(\top_h, \top_d, \{mp_2\})$ and $(\top_h, \top_d, \{mp_3\})$, are generated. Let us consider the first elementary vector, $(uh, \top_d, \top_{mp})$, the immediate successors generate by *MMISP* are $(uh, ca, \top_{mp})$, $(uh, \top_d, \{mp_1\})$ and $(uh, \top_d, \{mp_2\})$. Now, for the vector $(uh, ca, \top_{mp})$ obtained in the previous level, further immediate successors $(uh, ca, \{mp_1\})$ and $(uh, ca, \{mp_2\})$ are generated. Similarly we obtain the following two vectors $(uh, ca, \{mp_{11}\})$ and $(uh, ca, \{mp_1, mp_2\})$ and $(uh, ca, \{mp_{11}, mp_2\})$ from $(uh, ca, \{mp_1\})$ and $(uh, ca, \{mp_{11}\})$ respectively. Finally, for the vector $(uh, ca, \{mp_{11}, mp_2\})$, no any new frequent elementary vectors can be found, thus the generation stops.

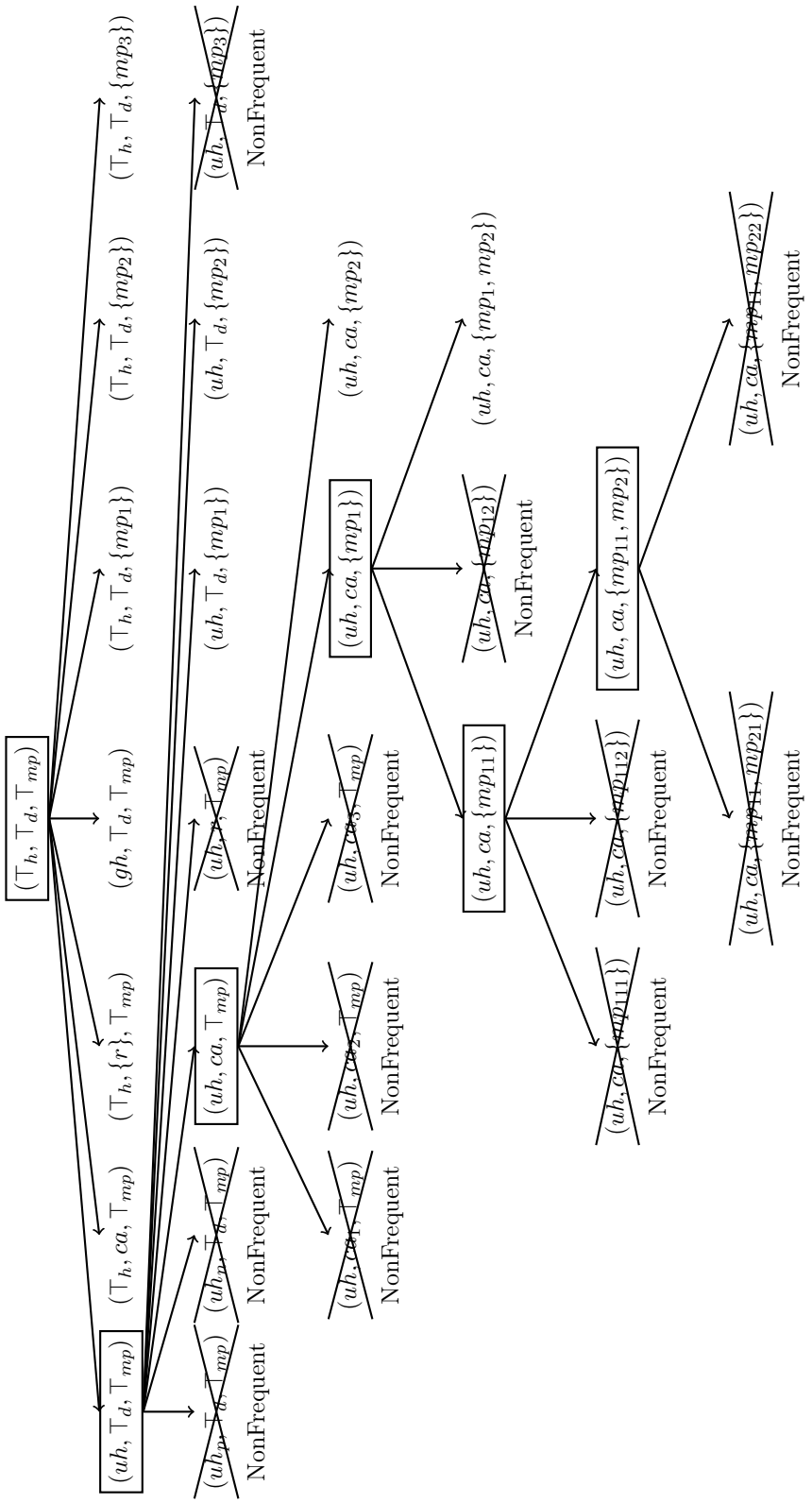


Figure 2.3: The steps of generating the elementary vectors in $(FE, \leqslant)$ with $\sigma = 3$.

id	Elementary Vector
1	$(uh, ca, \{mp_{11}, mp_2\})$
2	$(gh, r, \{mp_{22}, mp_{31}\})$
3	$(\top_h, \top_d, \{mp_{222}\})$

Table 2.2: The most specific frequent elementary vectors extracted from $(FE, \leqslant)$.

As the objective of *MMISP* is extracting the most specific sequential patterns, we retain only the most specific elementary vectors *MSFEV* in $(FE, \leqslant)$. The most specific frequent elementary vectors constitute collection of elementary vectors in $\mathcal{MS}_{\mathcal{DB}}$ with respect to σ which are frequent and most specific. Table 2.2 shows the set of most specific frequent elementary vectors which are extracted from $(FE, \leqslant)$.

The pseudocode of the first step in *MMISP* is presented in Algorithm 2. The idea is simple: Lines 2-3 generate the most general elementary vector $e = (\top_1, ..., \top_m)$, while Line 6 adds e to $(FE, \leqslant)$. Then, Line 7 recursively calls the function get_rec_msfev starting from e to build the *join-semilattice* $(FE, \leqslant)$ and to get all the most specific frequent elementary vectors $MSFEV$.

Algorithm 3 presents the pseudocode of the function get_rec_msfev. Lines 2-5 get the index z in e, Line 6 generates the immediate successor of e based on the index z, while Line 7 keeps only the frequent ones. When there are no frequent immediate successors of e, then e considers as one of the most specific frequent elementary vectors (see Lines 8-9). Otherwise, we call recursively the function get_rec_msfev for each frequent immediate successors of e (Lines 10-13).

Algorithm 4 shows how to generate immediate successors of an elementary vector. Line 2 shows that generation starts from the zth element. Lines 3-9 show that if an element is an item, we substitute it with one of its immediate successors and the rest of the elements are kept as it is. While, if an element is an itemset c, we substitute it with another itemset. This itemset is generated by substituting the last item in c with one of its immediate successors (Lines 13-17). Then, by appending the right sibling of last item in c at its end (Lines 18-22). Finally, by appending the right siblings of all the ancestors of last item in c at its end (Lines 23-29).

Algorithm 2: Mining All The Most Specific Frequent Elementary Vector

input : mds-database $\mathcal{MS}_{\mathcal{DB}}$, Minimum support threshold σ, Set of m taxonomies $Tax = \{Tax_1, ..., Tax_m\}$.

output: The set MSFEV of all most specific frequent elementary vectors, The join semi-lattice FE.

```
1 begin
    /* Generating the most general elementary vectors (⊤1, ⊤2, ..., ⊤m)   */
2   for i ← 1 to m do
3     e_i=⊤_i;
4   MSFEV ← ∅ ;
5   FE ← ∅ ;
6   FE ← FE ∪{ e } ;
    /* The recursive method to build the join-semilattice (FE, ⩽) and to get
       all the most specific frequent elementary vectors MSFEV.   */
7   call get_rec_msfev(MS_DB,e,σ,Tax,FE,MSFEV);
```

Algorithm 3: Routine get_rec_msfev

input : mds-database $\mathcal{MS}_{\mathcal{DB}}$, Elementary vector e, Minimum support threshold σ, Set of m taxonomies $Tax = \{Tax_1, ..., Tax_m\}$, Set of frequent elementary vectors FE, Set of most specific elementary vectors $MSFEV$.

```
1  begin
       /* Computing the index z                                          */
2      z =1;
3      for i ← 1 to m do
4          if e_i ≠ ⊤_i then
5              z =i;
6      Cand ← immediate_successor_elementary_vector(e,z,Tax) ;
7      Freq ← { e' ∈ Cand ; supp(e', MS_DB) ⩾ σ};
8      if Freq = ∅ then
           /* e is a most specific elementary vector                     */
9          MSFEV ← MSFEV ∪{ e } ;
10     else
           /* Recursive call get_rec_msfev for each new frequent elementary
              vector generated                                           */
11         foreach e' ∈ Freq do
12             FE ← FE ∪{ e' } ;
13             call get_rec_msfev(MS_DB,e,σ,Tax,FE,MSFEV);
```

2.3.2 Step 2: Transformation of mds-database:

We now study the temporal relation between the extracted specific frequent elementary vectors as follow. Firstly, we replace each elementary vector in each md-sequence of $\mathcal{MS}_{\mathcal{DB}}$ with all its generalizations from *MSFEV* set. Given a sequence $s = \langle s_1, ..., s_n \rangle$ in $\mathcal{MS}_{\mathcal{DB}}$ the replacement consists in substituting each elementary vector s_i in s by several elementary vectors $e \in MSFEV$ such that $s_i \leqslant e$.

Example 18. *Given the sequence s_4 in $\mathcal{MS}_{\mathcal{DB}}$ (see Table 2.1). The sequence s_4 is transformed into $\langle\{(uh, ca, \{mp_{11}, mp_2\}), (\top_h, \top_d, \{mp_{222}\})\}, \{(gh, r, \{mp_{22}, mp_{31}\})\}, \{(gh, r, \{mp_{22}, mp_{31}\})\}\rangle$ where:*

- *The elementary vector s_{41}, $(uh_p, ca_2, \{mp_{112}, mp_{222}\})$, is replaced by $(uh, ca, \{mp_{11}, mp_2\})$ and $(\top_h, \top_d, \{mp_{222}\})$ from* MSFEV *set in Table 2.2, with $s_{41} \leqslant (uh, ca, \{mp_{11}, mp_2\})$ and $s_{41} \leqslant (\top_h, \top_d, \{mp_{222}\})$.*

- *The elementary vector s_{42} and s_{43}, $(gh_p, r_2, \{mp_{221}, mp_{312}\})$, are replaced by $(gh, r, \{mp_{22}, mp_{31}\})$ from* MSFEV *set.*

Table 2.3 shows the transformation of $\mathcal{MS}_{\mathcal{DB}}$ in Table 2.1 based on the set of all most specific frequent elementary vectors *MSFEV* in Table 2.2. Transformation of $\mathcal{MS}_{\mathcal{DB}}$ denoted by $\widehat{\mathcal{MS}}_{\mathcal{DB}}$.

Algorithm 4: Routine *immediate_successor_elementary_vector*

input : Elementary vector e, Index z, Set of m taxonomies $Tax = \{Tax_1, ..., Tax_m\}$.
output: A set of immediate successors of e.

```
1  begin
2      for i ← z to m do
           /* In cas the element e_i is an item                          */
3          if e_i is an item then
4              foreach x ∈ immediate_successor (e_i,Tax_i) do
5                  e' =e;
6                  e'_i=x ;
7                  Output ← Output ∪ { e' } ;
8              end
9          end
           /* In cas the element e_i is an itemset                       */
10         else if e_i is an itemset then
11             c = e_i;
12             k = |c| ;
               /* Replacing the last item in c with its immediate successor */
13             foreach x' ∈ immediate_successor (c_k,Tax_i) do
14                 e' =e;
15                 e'_i = append_at_end(c\c_k, x') ;
16                 Output ← Output ∪ { e' } ;
17             end
               /* Appending the right sibling of c_k at the end of c      */
18             foreach x' ∈ right_sibling(c_k, Tax_i) do
19                 e' =e;
20                 e'_i = append_at_end(c, x') ;
21                 Output ← Output ∪ { e' } ;
22             end
               /* Appending the right sibling of all the ancestors of c_k at the
                  end of c                                                */
23             foreach x ∈ ancestors(c_k, Tax_i) do
24                 foreach x' ∈ right_sibling(x, Tax_i) do
25                     e' =e;
26                     e'_i = append_at_end(c, x') ;
27                     Output ← Output ∪ { e' } ;
28                 end
29             end
30
31     end
32     return Output ;
33 end
```

Patients	Trajectories
$\widehat{s}_1$	$\langle\{(uh, ca, \{mp_{11}, mp_2\})\}, \{(\top_h, \top_d, \{mp_{222}\})\}, \{(gh, r, \{mp_{22}, mp_{31}\})\}\rangle$
$\widehat{s}_2$	$\langle\{(uh, ca, \{mp_{11}, mp_2\})\}, \{(\top_h, \top_d, \{mp_{222}\}), (gh, r, \{mp_{22}, mp_{31}\})\}\rangle$
$\widehat{s}_3$	$\langle\{(uh, ca, \{mp_{11}, mp_2\})\}, \{(\top_h, \top_d, \{mp_{222}\}), (gh, r, \{mp_{22}, mp_{31}\})\}\rangle$
$\widehat{s}_4$	$\langle\{(uh, ca, \{mp_{11}, mp_2\}), (\top_h, \top_d, \{mp_{222}\})\}, \{(gh, r, \{mp_{22}, mp_{31}\})\}, \{(gh, r, \{mp_{22}, mp_{31}\})\}\rangle$

Table 2.3: A mds-database $\widehat{\mathcal{MS}}_{\mathcal{DB}}$ which is the transformation of the patient trajectories in Table 2.1 by using the set of all most specific frequent elementary vector in Table 2.2.

Patients	Trajectories
$\widehat{s}_1$	$\langle\{1\}, \{3\}, \{2\}\rangle$
$\widehat{s}_2$	$\langle\{1\}, \{2, 3\}\rangle$
$\widehat{s}_3$	$\langle\{1\}, \{2, 3\}\rangle$
$\widehat{s}_4$	$\langle\{1, 3\}, \{2\}, \{2\}\rangle$

Table 2.4: Transformed database in Table 2.3

2.3.3 Step 3: mds-patterns mining:

In a classical sequential pattern mining algorithm, the sequential database to be mined should be represented as a set of pairs (sid, s) where sid is a unique sequence identifier and s is a sequence of itemsets. To apply this algorithm on $\widehat{\mathcal{MS}}_{\mathcal{DB}}$, we transformed it as follows:

- Each elementary vector in *MSFEV* is assigned a unique *id* which is used during the mining (see Table 2.2).
- For each sequence $\widehat{s}_i$ in $\widehat{\mathcal{MS}}_{\mathcal{DB}}$ and for each elementary vector e in $\widehat{s}_{ij}$ where $\widehat{s}_{ij} \in \widehat{s}_i$, e is replaced by its *id*.

Example 19. *The sequence* $\widehat{s}_4 = \langle\{(uh, ca, \{mp_{11}, mp_2\}), (\top_h, \top_d, \{mp_{222}\})\}, \{(gh, r, \{mp_{22}, mp_{31}\})\}, \{(gh, r, \{mp_{22}, mp_{31}\})\}\rangle$ *in* $\widehat{\mathcal{MS}}_{\mathcal{DB}}$ *(see Table 2.3) is transformed into* $\langle\{1, 3\}, \{2\}, \{2\}\rangle$ *as:* $(\{uh\}, \{ca\}, \{mp_{11}, mp_2\})$, $(\{gh\}, \{r\}, \{mp_{22}, mp_{31}\})$ *and* $(\top_h, \top_d, \{mp_{222}\})$ *in* $\widehat{s}_4$ *has id 1, 2 and 3 respectively in Table 2.2.*

Table 2.4 shows the transformation of the database $\widehat{\mathcal{MS}}_{\mathcal{DB}}$ in Table 2.3 by using the identifiers of all most specific frequent elementary vectors *MSFEV* in Table 2.2.

We use CloSpan Yan et al. [2003] as a sequential pattern mining algorithm to extract sequential patterns form Table 2.4. Table 2.5 displays all sequential patterns in their transformed format and the frequent patient trajectories in which identifiers are replaced with their actual values, with $\sigma = 3$.

These 3 steps allow us to extract from heterogeneous multidimensional sequential database patterns that include elements with different levels of granularity.

2.4 Implementation and Experimental Validation

In this section, we conduct experiments on synthetic data that allowed us to study the scalability of the *MMISP* approach. We consider the number of extracted patterns and the running time

sequential patterns	mds-patterns	support
$\langle\{3\}\rangle$	$\langle(\top_h, \top_d, \{mp_{222}\})\rangle$	4
$\langle\{1\},\{2\}\rangle$	$\langle(uh, ca, \{mp_{11}, mp_2\}), (gh, r, \{mp_{22}, mp_{31}\})\rangle$	4
$\langle\{1\},\{3\}\rangle$	$\langle(uh, ca, \{mp_{11}, mp_2\}), (\top_h, \top_d, \{mp_{222}\})\rangle$	3

Table 2.5: All the most specific sequential patterns extracted from $\mathcal{MS}_{\mathcal{DB}}$ in Table 2.1 with $\sigma = 3$.

with respect several parameters. The MMISP algorithm is implemented in Java and the experiments are carried out on a MacBook Pro with a 2.5GHz Intel Core i5, 4GB of RAM Memory running OS X 10.6.8. Extraction of sequential patterns is based on the public implementation of CloSpan algorithm Yan et al. [2003] supplied by the IlliMine[2] toolkit.

2.4.1 Experiments on Synthetic Datasets

In this experiment, we study the scalability of the MMISP approach. We consider the number of extracted patterns and the running time with respect several parameters:

- number of elements in each elementary vector.
- depth of the taxonomy of each component.
- number of elementary vectors in each sequence (i.e. sequence length).
- number of sequences in a sequential database.

The first batch of synthetic data generated contains 1000 sequences defined over 2, 3, 4 and 5 elements in the elementary vector for each sequence. Each sequence contains 15 elementary vectors. Each taxonomy is defined over 3 levels of granularity between its items. Figure 2.4 reports the results according to different values of support threshold for different numbers of elements in the elementary vector. The running time increases for each newly added component.

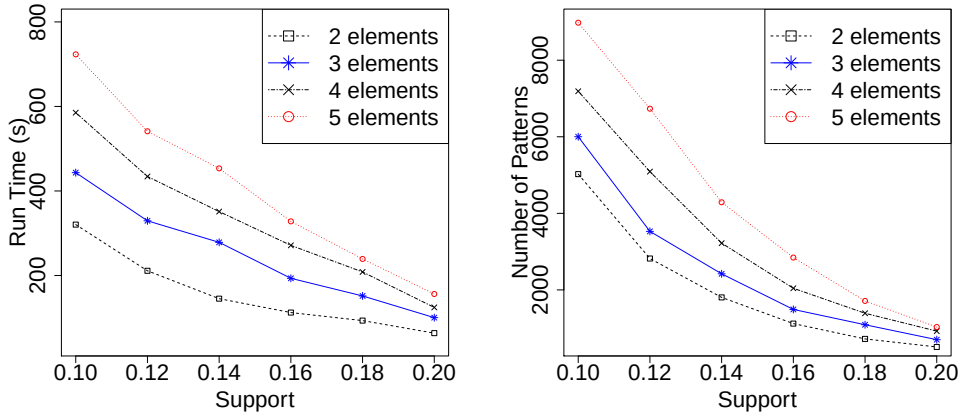
In Figure 2.5, we study the performance of *MMISP* by considering several levels of granularity for each taxonomy. We generated 1000 sequences defined over 15 elementary vectors. Each elementary vector has 3 components. Each taxonomy of the element is defined over 3, 4, 5, 6 levels of granularity. The number of extracted patterns does not change with each newly added level as *MMISP* extracts only the most specific sequential patterns. The execution time increases with the increase in the number of levels in a taxonomy because of the complexity of the product of taxonomies in the first step of *MMISP* which generates all the frequent elementary vectors.

We study the performance of *MMISP* and the number of extracted patterns with respect to the number of sequences in a sequential database and the length of each sequence. Figure 2.6 shows the execution time and the number of patterns extracted for 1000 sequences with 3 dimensions associated with taxonomy with 3 levels of granularity and with varying sequence length. The execution time increases with the increase in length of a sequence. This is due to the third step of *MMISP* which uses *CloSpan* to mine the transformed sequences.

Figure 2.7 shows the running time and the number of patterns extracted for several number of sequences (1000, 2000, 3000, 4000 and 5000 sequences) also with 3 dimensions associated with taxonomy with 3 levels of granularity. In these experiments, the sequence length is roughly 15.

Figures 2.4 - 2.7 highlight the fact that MMISP is efficient in terms of runtime for a large panel of sequences with varying different parameters.

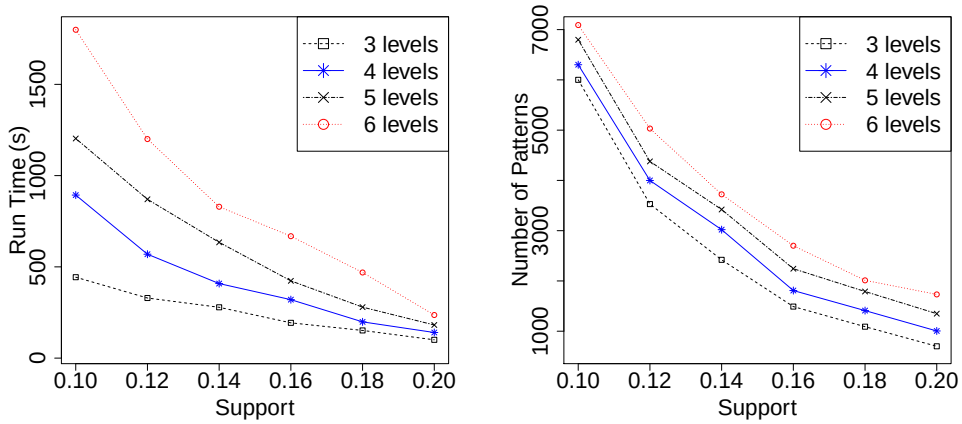
[2]http://illimine.cs.uiuc.edu/



(a) Runtime sequences over frequency threshold.

(b) Number patterns over frequency threshold.

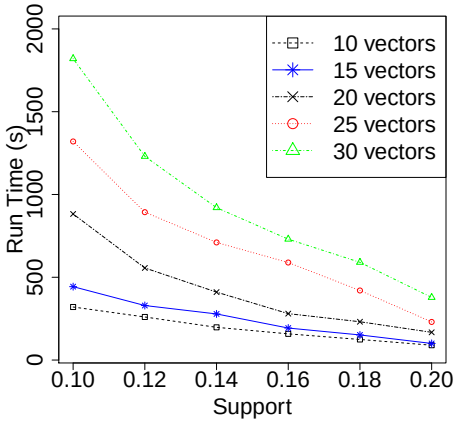
Figure 2.4: Number of sequential pattern extracted and Running time obtained by *MMISP* with varying in the length of elementary vectors.



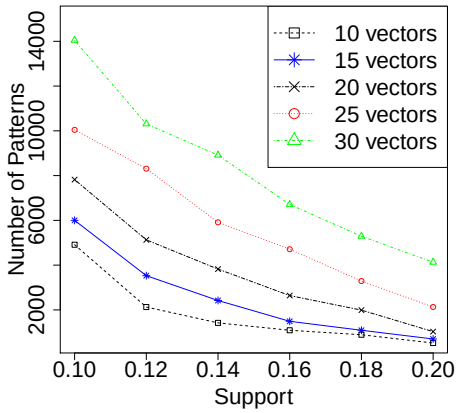
(a) Runtime sequences over frequency threshold.

(b) Number patterns over frequency threshold.

Figure 2.5: Number of extracted pattern (right) and Running Time (left) obtained by MMISP with varying over the levels of granularity between items of the taxonomies.

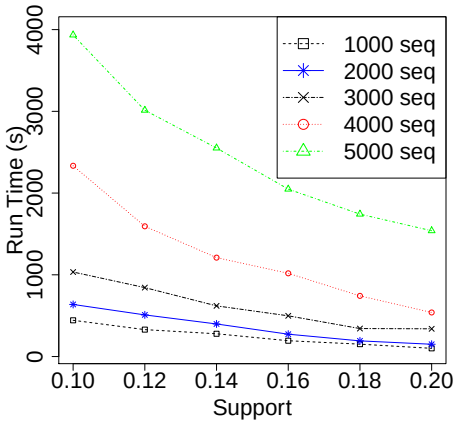


(a) Runtime sequences over frequency threshold.

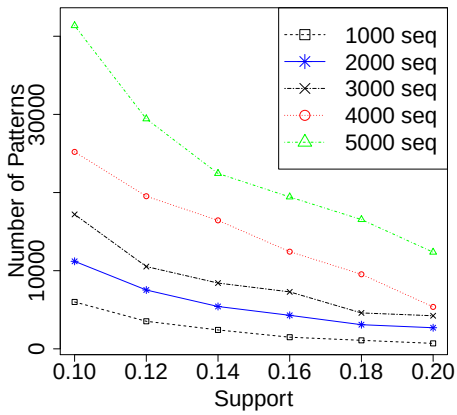


(b) Number patterns over frequency threshold.

Figure 2.6: Number of sequential pattern extracted and Running time for a large panel of sequences when varying over sequence length.



(a) Runtime sequences over frequency threshold.



(b) Number patterns over frequency threshold.

Figure 2.7: Number of sequential pattern extracted and Running time for a large panel of sequences when varying over several number of sequences.

2.5 Conclusion and perspective

In this chapter, we propose a new approach to mine heterogeneous multidimensional sequential patterns. Handling multidimensional hierarchies itemsets and sequential relations allows the discovery of patterns of the form $\langle(uh, ca, \{mp_{11}, mp_2\}), (gh, r, \{mp_{22}, mp_{31}\})\rangle$, defined on items and itemsets associated with taxonomies.

We define various concepts (elementary vector, md-sequence, mds-pattern and the most specific mds-pattern) and the approach *MMISP* for extracting the most specific multidimensional sequential patterns from a mds-database. Experiments conducted on a synthetic datasets highlight the fact that *MMISP* is efficient in terms of runtime for a large panel of sequences with varying parameters.

In our proposal, we define hierarchies as taxonomies. But in practice, hierarchies are generally directed graph with no directed cycles (DAG). Indeed, DAG (Directed Acyclic Graph) allows multiple hierarchies. The use of DAG in our proposal require some adjustments. In a DAG, there may be two different paths from the root to a node of the graph. It is important to note that each path is associated with a specific semantic. Integration of hierarchies as DAGs must take into account the semantics of each path. Thus, a preferred path relative to another can help to prune faster the search space and adapt easily the algorithm MMISP.

We are aware that choosing the most specific frequent elementary sequence to mine the sequential patterns prevents us from extracting all of the most specific sequential patterns. Coping with this issue is another interesting extension of the present work that we plan to investigate in the future.

We are also planning to use statistical significance tests to evaluate the sequential patterns extracted and choose the most significant ones. On the other hand, proposing a graphical interface to visualize and query the sequential patterns.

Chapter 3

On Projections of Sequential Pattern Structures

Contents

3.1 Introduction . . . 53
3.2 Formal Concept Analysis . . . 55
3.2.1 Formal Context . . . 55
3.2.2 Formal Concept . . . 55
3.2.3 Concept Lattice . . . 57
3.2.4 Algorithms . . . 59
3.3 Reducing the Representation Complexity of Concept Lattice . . . 59
3.3.1 Iceberg Concept Lattice . . . 59
3.3.2 Stability . . . 60
3.4 Summarizing Closed Sequential Patterns with Partial Order . . . 62
3.5 Classification and selection of interesting sequential patterns by using FCA . . . 64
3.6 Multidimensional Sequential Pattern Structures . . . 67
3.6.1 Pattern Structures . . . 67
3.6.2 Sequential Data . . . 67
3.6.3 Multidimensional Sequential Meet-semilattice . . . 68
3.7 Projections of Sequential Pattern Structures . . . 70
3.8 Conclusion . . . 71

3.1 Introduction

Formal Concept Analysis (FCA) is a mathematical framework which allows data analysis and knowledge representation. FCA was first introduced in early 1980s. FCA is actually based on lattice and order theory first introduced by Birkhoff in Birkhoff [1964]. Then, Galois lattices were introduced in 1970s by Barbut and Monjardet [1970]. In 1982, FCA was introduced under this name by Wille [1982]. FCA was first developed by the community of mathematicians where it was conceived as reconstruction of general algebra and lattice theory. From 1990, FCA has successfully been applied in many domains, such as medicine and psychology Jay

et al. [2013], musicology Motoyoshi et al. [2009], linguistic databases Priss [2005], information science Priss [2006], software engineering Tilley et al. [2005], knowledge representation Guarino [1995], information retrieval Codocedo et al. [2014] and more recently knowledge extraction from data Valtchev et al. [2004]. A strong feature of Formal Concept Analysis is its capability of producing graphical visualizations present in the form of concept hierarchies. Most researched area deals with limitation in visualizing concept lattice for large dataset. To overcome this limitation, Stumme et al. [2002] had introduced a notion of concept frequency which considers only the frequent concepts in the concept lattice called iceberg concept lattice. Kuznetsov [1990], Kuznetsov et al. [2007] also have introduced stability as a new interesting measure for checking the stability of a concept w.r.t the whole lattice. Stability has been successfully used for pruning concept lattices and choosing only the stable concepts.

In chapter 2 we presented *MMISP*, an approach for mining heterogeneous multidimensional sequential patterns. However, the problem with *MMSIP* and all multidimensional sequential pattern mining algorithms is that comparable elements do not appear in the discovered sequential patterns. This limitation does not allow the discovery of patterns going from precise to general knowledge or vice versa. In this chapter, we focus on complementing the multidimensional sequential pattern mining approaches with a sound and adequate algebraic approach. That is, *can we develop a framework for enumerating patterns that describe the inner trends by displaying patterns that either go from precise knowledge to general knowledge or go from general knowledge to precise knowledge?*.

The above question can be solved by addressing the problem of mining sequential data with pattern structures Ganter and Kuznetsov [2001b], an extension of FCA that handles complex data. Pattern structures can be hard to process usually due to the large number of concepts in the concept lattice and the complexity of the involved descriptions and the similarity operation. To solve this problem, we introduce and discuss the notion of *projections* which are mathematical functions that respect certain algebraic properties. These mathematical objects significantly decrease (i.e., filter) the number of patterns, while preserving the most interesting ones for an expert. Moreover, the projection provides an efficient tool for the analysis of multidimensional sequential datasets.

In this chapter, we firstly use FCA as a lattice-based classification method to regroup the multidimensional sequential patterns mined with MMISP. We show the interesting properties of concept lattices and stability index to classify and select interesting patterns. Then, we propose a novel way of mining multidimensional sequences by mapping them to pattern structures. The genericity power provided by the pattern structures allows our approach to be directly instantiated with state-of-the-art FCA algorithms, making the final implementation flexible, accurate and scalable. Then, we introduce and discuss the notion of *"projections"* for sequential pattern structures. The advantage of projections is its ability to significantly decrease the complexity of a problem, saving thus computational time and preserving the most interesting ones for an expert.

The remaining of this chapter is organized as follows. Section 3.2 is dedicated to the theoretical foundations of the Formal Concept Analysis. Section 3.3 introduces the definition of iceberg concept lattice and stability measure. Section 3.4 presents the method exist to generate partial orders out of the set of closed sequences. Section 3.5 presents how we can use Formal Concept Analysis to group the multidimensional sequential patterns. Section 3.6 presents a novel way of dealing with multidimensional sequences by mapping them to pattern structures. In Section 3.7, we introduce and discuss the notion of *projections* for sequential pattern structures. Section 3.8 concludes this chapter

	diabetes	hypertension	dyslipidemia	anger
p_1	×	×		
p_2		×		
p_3		×		
p_4	×		×	
p_5	×		×	×
p_6	×			
p_7	×	×	×	×
p_8	×	×		

Table 3.1: Formal context of pathology

3.2 Formal Concept Analysis

This section discusses basic notions, definitions and proprieties related for FCA based on Ganter and Wille [1999].

3.2.1 Formal Context

Definition 34. *(Formal Context)*
A formal context $\mathbb{K} = (G, M, I)$, consists of two sets G and M and a binary relation I between G and M. Elements of G are called objects while elements of M are called attributes of the context. The fact $(g, m) \in I$ is interpreted as "the object g has attribute m"*. A formal context can be viewed in a binary table.*

Example 20. *Table 3.1 presents the formal context related to pathology characterizing a series of patients and their pathologies. The patients are kept as a set of objects in the formal context while the pathologies are considered as the attributes. The presence of a disease in a patient defines a relationship I and is marked by a cross at the intersection of their column and row. According to first row in the formal context, patient p_1 has diabetes and hypertension.*

3.2.2 Formal Concept

In a formal context, a formal concept represents a natural groupings of objects based on their shared properties. For defining the notion of formal concept, we introduce the following operators.

Definition 35. *(Derivation Operators)*
Let $\mathbb{K} = (G, M, I)$ be a formal context. For a set of objects $A \subseteq G$, we define the set of attributes that all objects in A have in common as follows:

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

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

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

Definition 36. *(Composition Operator)*
Let $\mathbb{K} = (G, M, I)$ be a formal context. For a set of objects $A \subseteq G$ and a set of attributes $B \subseteq M$, we define the composition operator, denoted by $''$, which map the set of objects onto itself and the set of attributes onto itself as follows:

$$A'' = (A')'$$

$$B'' = (B')'$$

Example 21. *Consider the formal context in Table 3.1. we have:*

$\{p_1,\ p_8\}' = \{diabetes,\ hypertension\}$

$\{diabetes,\ dyslipidemia,\ anger\}' = \{p_5,\ p_7\}$

$\{diabetes,\ hypertension\}$ represents the set of pathologies that are common to the patients p_1 and p_8. Similarly, $\{p_5,\ p_7\}$ represents the set of patients who suffer diabetes, dyslipidemia and anger. An example of a composition operator is:

$\{p_1,\ p_8\}'' = \{diabetes,\ hypertension\}' = \{p_1,\ p_7,\ p_8\}$

$\{diabetes,\ dyslipidemia,\ anger\}'' = \{p_5,\ p_7\}' = \{diabetes,\ dyslipidemia,\ anger\}$

Definition 37. *(Formal concept)*
Let $\mathbb{K} = (G, M, I)$ be a formal context. A formal concept of a context $\mathbb{K}$ is a pair C=(A,B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. A is called the extent of the concept C while B is called its intent. The set of all formal concepts of a context $\mathbb{K}$ is written $\mathfrak{B}$(G,M,I).

We notice that if C=(A,B) is a formal concept, by definition we have: $A'' = A$ and $B'' = B$.

Definition 38. *(Subconcept, Superconcept)*
Let $\mathbb{K} = (G, M, I)$ be a formal context and $\mathfrak{B}$(G,M,I) be the set of all formal concepts of a context $\mathbb{K}$. Let $C_1 = (A_1, B_1)$ and $C_2 = (A_2, B_2)$ be two concepts in $\mathfrak{B}$(G,M,I). The concept C_1 is a subconcept of C_2, or that C_2 is a superconcept of C_1, denoted by $C_1 \leqslant C_2$, if and only if:

$$A_1 \subseteq A_2$$

Property 5. *Let $C_1 = (A_1, B_1)$ and $C_2 = (A_2, B_2)$ be two concepts in $\mathfrak{B}$(G,M,I).*

$$C_1 \leqslant C_2 \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow B_2 \subseteq B_1$$

Example 22. *Consider the formal context in Table 3.1. The pair $(\{p_5, p_7\},$ {diabetes, dyslipidemia, anger}) is a formal concept where {diabetes, dyslipidemia, anger}$'$=$\{p_5,\ p_7\}$ and $\{p_5,\ p_7\}'$={diabetes, dyslipidemia, anger}. The set of pathologies that are common to the patients p_5 and p_7 are diabetes, dyslipidemia and anger. All patients who have both diabetes, dyslipidemia and anger are p_5 and p_7. An example of $\leqslant$-relation between two concepts is given by:*

$$(\{p_5, p_7\},\ \{diabetes,\ dyslipidemia,\ anger\}) \leqslant (\{p_4, p_5, p_7\},\ \{diabetes,\ dyslipidemia\})$$

On the other hand, $(\{p_7, p_8\}$, {diabetes, hypertension, dyslipidemia}) is not formal concept because of:

$\{p_7, p_8\}' = \{diabetes,\ hypertension\}$

	diabetes	dyslipidemia	anger	hypertension
p_5	×	×	×	
p_7	×	×	×	×
p_1	×			×
p_2				×
p_3				×
p_4	×		×	
p_6	×			
p_8	×			×

Table 3.2: The concept $(\{p_5, p_7\},\ \{diabetes,\ dyslipidemia,\ anger\})$ is a maximum rectangle.

$$\{diabetes,\ hypertension,\ dyslipidemia\}' = \{p_7\}$$

Formal concept consists of a set of objects sharing some attributes. Consider a concept C(A,B) where A is the set of objects called *extent* sharing some set of attributes called *intent*. The definition of a formal concept meets a principle of maximality and it is not possible to extend the extent of a concept without changing its intent and vice versa. This can be visualized in a formal context by interchanging the rows and column of the formal context. A formal concept appears as a rectangle filled with maximum cross. Table 3.2 shows the concept $(\{p_5, p_7\},\ \{diabetes,\ dyslipidemia,\ anger\})$ in the formal context of pathology as a maximum rectangle.

3.2.3 Concept Lattice

The set of concepts obtained from a formal context, with the order relation of the concepts, has a specific mathematical structure called *Concept Lattice*. Before defining the notion of concept lattice, we recall the following definitions.

Definition 39. *(Ordered set)*
Given an order relation $\leqslant$ on a set M, an ordered set is a pair $(M, \leqslant)$. When $\leqslant$ is a partial order, $(M, \leqslant)$ is called partially ordered set, or poset.

Definition 40. *(Infimum, Supremum)*
Let $(M, \leqslant)$ be an ordered set and A a subset of M. A lower bound of A is an element a of M with $a \leqslant b$ for all $b \in A$. An upper bound of A is defined dually. If it exists a largest element in the set of all lower bounds of A, it is called the infimum of A and is denoted by "inf A" *or $\bigwedge A$; dually, a least upper bound is called supermum and denoted by* "sup A" *or $\bigvee A$. Infimum and supremum are frequently called respectively meet and join, also denoted respectively by the symbols $\sqcap$ and $\sqcup$.*

Definition 41. *(Lattice)*
A poset $\mathcal{V} = (V, \leqslant)$ is a lattice, if for any two elements $x, y \in V$ the supermum $x \vee y$ and the infimum $x \wedge y$ always exist.

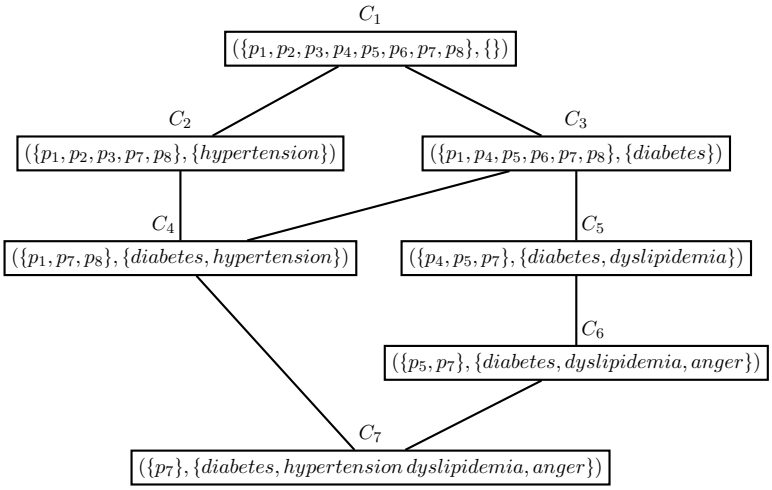


Figure 3.1: Concept lattice generated from Table 3.1

Definition 42. *(Complete lattice)*
A poset $\mathcal{V} = (V, \leqslant)$ is a complete lattice, if for any subset $X \subseteq V$ the supermum $\bigvee X$ and the infimum $\bigwedge X$ exist.

The set of all formal concepts from a formal context $\mathbb{K}$ order with the relation $\leqslant$ form a complete lattice called *concept lattice* of $\mathbb{K}$ and denoted by $\underline{\mathfrak{B}}$(G,M,I).

The Basic Theorem on Concept Lattice shows that a concept lattice is complete and defines its infimum and supermum.

Theorem 1. *(The Basic Theorem on Concept Lattice)*
The concept lattice $\underline{\mathfrak{B}}(G,M,I)$ is a complete lattice in which infimum and supermum are given by:

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

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

The supremum of all the concepts of $\underline{\mathfrak{B}}$(G,M,I) is denoted by $\top$ (top) and their infimum is denoted by $\bot$ (bottom).

Example 23. *Figure 3.1 shows a concept lattice associated to Table 3.1. Here, each node denotes a concept while an edge denotes an order relation between two concepts. The top concept (resp. bottom concept) is the highest (resp. lowest) w.r.t $\leqslant$. C_1 is the top concept ($\top$) while C_7 is the bottom concept ($\bot$).*

3.2.4 Algorithms

Several algorithms have been proposed for the construction of concept lattice: *Close by one* Kuznetsov [1993], *Chein* Chein [1969], *Norris* Norris [1978], *NextClosure* Ganter [2010], Ganter and Reuter [1991], *Bordat* Bordat [1986], etc. Kuznetsov and Obiedkov [2002] focus on their theoretical complexities and an experimental comparison on several datasets. The authors make their recommendations depending on the nature of the context.

In addition, several applications have been developed to build, manipulate and visualize Concept Lattice. *Coron*[3], *ConExp*[4] and *Galicia*[5] are the open source tools commonly used. These applications are equipped with all the basic functionality for building and manipulating a concept lattice i.e., editing formal context, constructing and visualizing a concept lattice. Coron is one of the platforms that helps in building iceberg lattices given a certain support threshold with the help of several algorithms. It also helps in generating association rules using several algorithms which can be filtered with the help of support and confidence.

3.3 Reducing the Representation Complexity of Concept Lattice

3.3.1 Iceberg Concept Lattice

A well known problem with FCA is a limitation in the visualization concept lattices for large and dense formal contexts as it generates huge lattices consisting of thousands of concepts. Stumme et al. [2002] propose an algorithm *TITANIC* to overcome this limitation by visualizing only the frequent concepts in the concept lattice called iceberg concept lattice.

Definition 43. *(Support of Concept)*
Let $\mathbb{K} = (G, M, I)$ be a formal context and $C = (A, B)$ be a formal concept of the context $\mathbb{K}$. The support of concept is defined as follows:

$$\sigma(C) = \frac{|A|}{|G|}$$

Definition 44. *(Frequent Concept)*
Let $\mathbb{K} = (G, M, I)$ be a formal context, $C = (A, B)$ be a formal concept of the context $\mathbb{K}$ and σ be a minimal support threshold. The concept C is called frequent concept if $\sigma(C) \geqslant \sigma$.

Definition 45. *(Iceberg Concept)*
Let $\mathbb{K} = (G, M, I)$ be a formal context and σ be a minimal support threshold. The set of all frequent concepts of a context $\mathbb{K}$ is called iceberg concept of the context $\mathbb{K}$.

Property 6. *An iceberg concepts has a structure of join-semi-lattice.*

This property follows from the monotony of the support function. The support function is monotonously decreasing: given two attribute sets of context $\mathbb{K}$; $B_1, B_2 \subseteq M$; $B_1 \subseteq B_2 \Rightarrow \sigma(B_1) \leqslant \sigma(B_2)$. Thus, an iceberg concept lattice is an order filter of the whole concept lattice and in general only a join-semi-lattice. Meanwhile, adding a bottom element, $\perp$, makes it a lattice again.

Example 24. *Figure 3.2 shows the iceberg concept lattice associated with Table 3.1 with a minimal support threshold equals to 0.35. If we set a support threshold equals 0.6, the iceberg concept lattice will contain only three concepts, i.e., C_1, C_2 and C_3.*

[3]http://coron.loria.fr
[4]http://sourceforge.net/projects/conexp
[5]http://www.iro.umontreal.ca/ galicia/

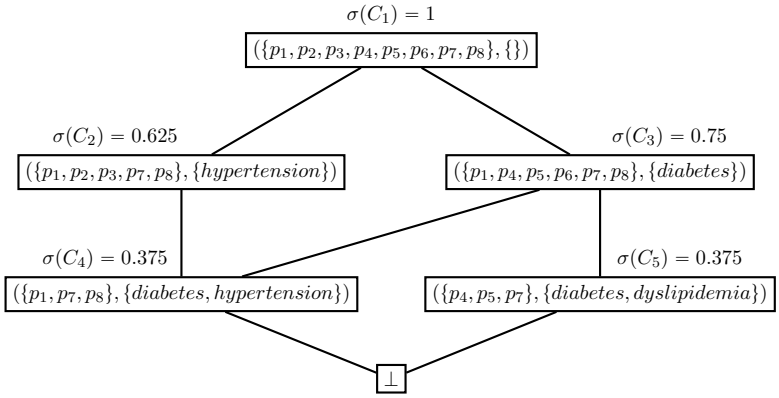


Figure 3.2: Concept lattice raised from Table 3.1 with a minimal support threshold equals to 0.35

3.3.2 Stability

The notion of stability of a formal concept was first introduced in 1990 by Kuznetsov [1990]. Since then, other definitions have been given by the same author Kuznetsov et al. [2007], Kuznetsov [2007]. Here, we rely on the definition given in Kuznetsov et al. [2007].

Definition 46. *(Stability of concept)*
Let $\mathbb{K} = (G, M, I)$ be a formal context and $C = (A, B)$ be a formal concept of the context $\mathbb{K}$. The stability of concept is defined as follows:

$$\gamma(C) = \frac{|X \subseteq A;\ X^{'} = A^{'} = B|}{2^{|A|}}$$

The stability index of a concept indicates how the intent of a particular concept depends on the extent of that concept. It indicates the probability of preserving concept intent while removing some objects of its extent. A stable concept continues to be a concept even if a few objects are removed. This means also that a stable concept is resistant to noise and will not collapse when some members are removed from its extent.

Given a concept $C = (A, B)$, the stability index measures the number of elements of G that are in the same *equivalence class* of A, where an equivalence class is defined as follows.

Definition 47. *(Equivalence Class)*
Let $\mathbb{K} = (G, M, I)$ be a formal context and $X \subseteq G$. The equivalence class of X, denoted by $\langle X \rangle$, is defined as follows:

$$\langle X \rangle = \{Y \subseteq G;\ Y^{'} = X^{'}\}$$

Thus, considering a formal concept $C = (A, B)$, definition of stability can be rewritten as:

$$\gamma(C) = \frac{|\langle A \rangle|}{2^{|A|}}$$

Computing stability is a #P-complete problem Kuznetsov et al. [2007]. Meanwhile, once the concept lattice has been computed, a bottom-up traversal algorithm can efficiently compute stability Roth et al. [2008]. Actually, a concept stability depends on the stability of its subconcepts Jay et al. [2008]. This can be shown as follows:

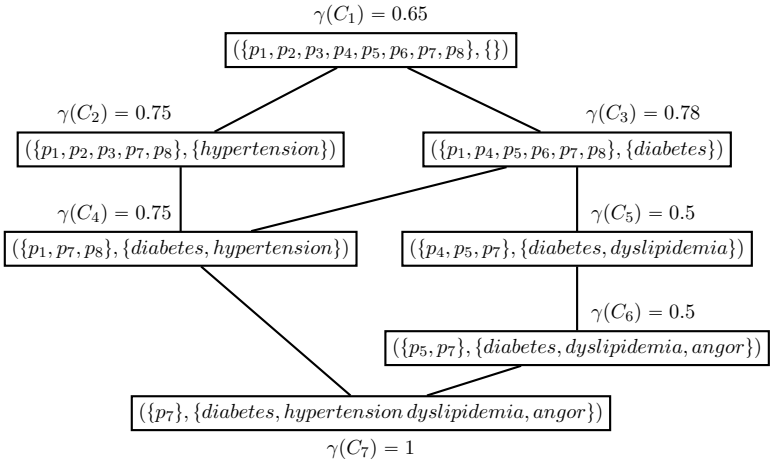


Figure 3.3: Example of calculation of stability in a lattice in Figure 3.1

Property 7. *Let $C_1 = (A_1, B_1)$ be a formal concept of the context (G, M, I).*

$$\gamma(C_1) = 1 - \sum_{\substack{C_2=(A_2,B_2);\\ A_2 \subset A_1 \text{ and } A_2 = A_2''}} \gamma(C_2) \times 2^{|A_2|-|A_1|}$$

Example 25. *The Figure 3.3 shows an example of calculation of stability in a concept lattice in the Figure 3.1. Each concept is labelled by its stability. Thus, for the concept $C_5 = (\{p_4, p_5, p_7\}, \{diabetes, dyslipidemia\})$, we have:*

$$\begin{aligned}
\varnothing' &= \{diabetes, hypertension, dyslipidemia, angor\} \neq \{diabetes, dyslipidemia\} \\
\{p_4\}' &= \{diabetes, dyslipidemia\} \\
\{p_5\}' &= \{diabetes, dyslipidemia, angor\} \neq \{diabetes, dyslipidemia\} \\
\{p_7\}' &= \{diabetes, hypertension, dyslipidemia, angor\} \neq \{diabetes, dyslipidemia\} \\
\{p_4, p_5\}' &= \{diabetes, dyslipidemia\} \\
\{p_4, p_7\}' &= \{diabetes, dyslipidemia\} \\
\{p_5, p_7\}' &= \{diabetes, dyslipidemia, angor\} \neq \{diabetes, dyslipidemia\} \\
\{p_4, p_5, p_7\}' &= \{diabetes, dyslipidemia\}
\end{aligned}$$

This gives $\gamma((\{p_4, p_5, p_7\}, \{diabetes, dyslipidemia\})) = \frac{4}{8} = 0.5$.

Stability offers an alternative point of view on concepts compared to the well known metric of support based on frequency. Jay et al. [2008] present that combining concept frequency together with concept stability provides a very efficient means for discovering and analyzing a social healthcare network in the field of cancer care. Jay et al. [2008] shows that using interesting measures for pruning the concept lattice has a strong impact both on readability and semantics of discovered knowledge. With support and stability, Jay et al. [2008] identify two kinds of concepts: frequent unstable concepts and rare stable concepts.

Example 26. *Figure 3.4 shows the concept lattice associated with Table 3.1 and with a minimal support threshold equals 0.3 and a minimal stability threshold equals 0.5.*

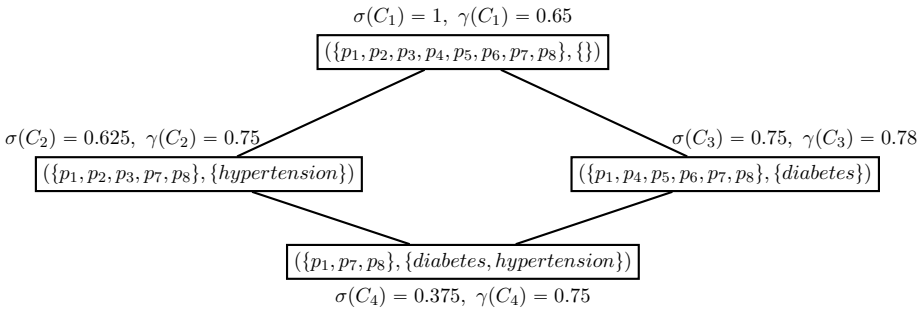


Figure 3.4: Concept lattice associated with Table 3.1 and with a minimal support threshold equals 0.3 and a minimal stability threshold equals 0.5

3.4 Summarizing Closed Sequential Patterns with Partial Order

FCA processes only binary context, which limits exploration of data represented in other complex forms. More precisely, FCA does not allow manipulation of sequential data or a multidimensional sequential data. The objective of applying FCA on the sequential data is grouping the sequences occurring together in a maximal set of transactions then constructing a partial order out of those groups. To our knowledge, there is only one proposal Casas-Garriga [2005], Balcázar and Casas-Garriga [2005] which summarizes closed sequential patterns extracted from sequential data with partial order.

Casas-Garriga [2005], Balcázar and Casas-Garriga [2005] use FCA theory to formalize a new closure system that characterizes sequential data. In this work, the authors try to generalize the idea of making closed sequential patterns as much compact as possible so that they can produce not just fewer patterns, but also more informative ones. They show how these closed sequences coexisting together lead to a novel notion of closed partial order that summarize the sequential data in a compact way.

The following definitions, propositions and theorems follow the two papers Casas-Garriga [2005], Balcázar and Casas-Garriga [2005].

Definition 48. *(Formal Context)*

A formal context $\mathbb{K} = (G, M, R)$ consists of two sets G and M and a relation R, where G is the set of objects, M is the set of items and $R \subseteq G \times M \times \mathcal{N}$. For an entry $(g, i, k) \in R$ we read as " the item i occurs at the position k in the object g"*; so, k represents the order of occurrence of i with respect to the other items in the same object.*

Example 27. *Figure 3.5 shows a set of sequential database and its associated context. Each sequence in $\mathcal{S}_{DB}$ can be represented as an object, so that the entry in row g_j and column i corresponds to the order/s of the item i in the equivalent input sequence s_j.*

Then, the authors define two *derivation operators* ϕ and ψ as follows:

Definition 49. *(Derivation Operators)*

Let $\mathbb{K} = (G, M, R)$ be a formal context. Two operators ϕ and ψ which connect the power sets

Seq id	Sequence
s_1	$\langle\{a,e\}\{c\}\{d\}\{a\}\rangle$
s_2	$\langle\{d\}\{a,b,e\}\{f\}\{b,c,d\}\rangle$
s_3	$\langle\{d\}\{a\}\{b\}\{f\}\rangle$

(a) Sequential Database $\mathcal{S}_{DB}$

Objects	a	b	c	d	e	f
g_1	1,4		2	3	1	
g_2	2	2,4	4	1,4	2	3
g_3	2	3		1		4

(b) Formal context of $\mathcal{S}_{DB}$

Figure 3.5: Example of sequential database and its ordered context

of objects 2^G *and sequences* $2^{\mathcal{S}}$ *are defined as follows:*

$$\phi : 2^G \longrightarrow 2^{\mathcal{S}} \quad , \quad \phi(G) = \{s \in \mathcal{S}; s \text{ maximally contained in } g, \forall g \in G\}$$
$$\psi : 2^{\mathcal{S}} \longrightarrow 2^G \quad , \quad \psi(S) = \{g_i \in G; s \subseteq s_i, \forall s \in S\}$$

The mapping $\phi(G)$ returns the set of sequences common to all the objects in G, while $\psi(S)$ returns the set of input sequences that include all the sequences in S.

Example 28. *Consider the context in Figure 3.5, we have that* $\phi(\{g_1, g_3\}) = \{\langle\{d\}\{a\}\rangle\}$*, and* $\psi(\{\langle\{a,e\}\{d\}\rangle, \langle\{a,e\}\{c\}\rangle) = \{g_1, g_2\}$

Proposition 1. *Composition* $\widehat{\Delta} = \psi \cdot \phi$ *and* $\Delta = \phi \cdot \psi$ *are closure operators.*

Definition 50. *Closed sets of sequences are those coinciding with their closure, that is:*

$$\Delta(S) = S$$

Proposition 2. *All sequences in a closed set are maximal in it w.r.t.* $\subseteq$.

Theorem 2. *Let S be a closed set of sequecnes, then for all* $s \in S$*, s is a closed sequence.*

Definition 51. *(Formal Concept)*
A formal concept is a pair (G, S) *where G is a set of objects, S is a set of sequences, and* $\phi(G) = S$ *and* $\psi(S) = G$*. Then, G is the extent and S is the intent of the concept, and both sets are said be linked by the Galois connection. The concepts are also called closed since we have that* $\Delta(S) = S$ *and* $\widehat{\Delta}(G) = G$*.*

Example 29. *Consider the context in Figure 3.5. The pair* $(\{g_1, g_2, g_3\}, \{\langle\{d\}\{a\}\rangle\})$ *is a formal concept where* $\phi(\{g_1, g_2, g_3\}) = \{\langle\{d\}\{a\}\rangle\}$ *and* $\psi(\{\langle\{d\}\{a\}\rangle\}) = \{g_1, g_2, g_3\}$*. Figure 3.6 shows the concept lattice associated with ordered context in Figure 3.5.*

Definition 52. *We say that a set of sequences* S_1 *is more general than another set of sequences* S_2*, denoted by* $S_1 \preceq S_2$*, if and only if for all* $s_1 \in S_1$*, there exists* $s_2 \in S_2$ *such that* $s_1 \subseteq s_2$*. Then* S_2 *is also said to be more specific than* S_1*.*

Definition 53. *Let* $C_1 = (G_1, S_1)$ *and* $C_2 = (G_2, S_2)$ *be two concepts. The concept* C_1 *is subconcept of* C_2 *if* $G_1 \subseteq G_2$ *(equivalent to* $S_2 \preceq S_1$*).*

Casas-Garriga [2005], Balcázar and Casas-Garriga [2005] assume that the set of closed sequences already exist which are mined by any of the existing algorithms like CloSpan Yan et al. [2003] and then these sequences are postprocessed. This postprocessing is done by comparing the object id list of the closed sequences and properly organizing them in a pair (G, S) so that each $s \in S$ has the same object id list. First, closed sequences are ordered in descending order

w.r.t number of object in its object id list, then the algorithm traverses the conceptual graph bottom-up in a breadth-first fashion. The grouping of mined closed sequences is performed as follows: for each close set of objects G, its closed sequences serve as an initial point for the construction of the intent S linked to G through the Galois connection; then, S is completed with the maximal sequences of immediate predecessors (see Algorithm 5).

Algorithm 5: Grouping Close Sequential Patterns into Formal Concepts

Input : List $\mathcal{CSP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s)$ of close sequential patterns extracted by CloSpan
Output: List of concepts $\mathfrak{B}$

```
1  begin
2      Sort CSP(S_DB, σ_s) in descending order by object id list ;
3      while CSP(S_DB, σ_s) ≠ ∅ do
4          S_1 ← Next sequence s_1 ∈ S_1 with same tid list ;
5          G_1 ← object_id(s_1), for some s_1 ∈ S_1 ;
6          foreach (G_2, S_2) immediate predecessor of (G_1, S_1) do
7              foreach s_2 ∈ S_2 do
8                  if s_2 ⋢ s_1, for each s_1 ∈ S_1 then
9                      S_1 ← S_1 ∪ {s_2} ;
10         B ← B ∪ {(G_1, S_1)} ;
11     Output B ;
```

Example 30. *Table 3.3 shows all closed sequential patterns in the sequential dataset $\mathcal{S}_{DB}$ (see Table 3.5.a). Concept lattice associated to the extracted closed sequential patterns is shown in Figure 3.6 and is generated by the Algorithm 5. For example, the formal concept $C_2 = (\{g_1, g_2\}, \{\langle\{a, e\}\{c\}\rangle, \langle\{a, e\}\{d\}\rangle, \langle\{d\}\{a\}\rangle\})$ is built by considering the object id list $\{g_1, g_2\}$ from Table 3.3 as its extent and the intent is built by grouping all the closed sequential pattern of $\{g_1, g_2\}$ in one set $\{\langle\{a, e\}\{c\}\rangle, \langle\{a, e\}\{d\}\rangle\}$. Then, the algorithm checks if there is any sequence in its immediate predecessor (i.e, C_1) which is not a subsequence of any sequence in the intent of C_2. If there is such a sequence, it is added to the intent of C_2. For example, the sequence $\langle\{d\}\{a\}\rangle$ in the intent of C_1 is not subsequence of either $\langle\{a, e\}\{c\}\rangle$ or $\langle\{a, e\}\{d\}\rangle$, so we add $\langle\{d\}\{a\}\rangle$ to the intent of C_2. Finally, we have $\{\langle\{a, e\}\{c\}\rangle, \langle\{a, e\}\{d\}\rangle, \langle\{d\}\{a\}\rangle\}$ as an intent of C_2.*

The algorithmic proposal is bounded by $O(n \cdot m \cdot k^2)$, where we consider n to be the number of closed sequences in $\mathcal{CSP}(\mathcal{S}_{\mathcal{DB}}, \sigma_s)$, m is the maximum number of immediate predecessors for a node (G, S), and k is the maximum number of closed sequences belonging to immediate predecessors of S. This approach, while theoretically sound, may become inefficient with a large set of sequences, and does not compute the stability for each formal concept.

3.5 Classification and selection of interesting sequential patterns by using FCA

As presented in the previous section, Casas-Garriga summarized closed sequential patterns extracted from sequential data with partial order. In this section, we will present another way of regrouping the sequential patterns with partial order by using a standard FCA.

Pattern id	Object id list	Closed sequential patterns
sp_1	$\{g_1, g_2, g_3\}$	$\langle\{d\}\{a\}\rangle$
sp_2	$\{g_1, g_2\}$	$\langle\{a, e\}\{c\}\rangle$
sp_3	$\{g_1, g_2\}$	$\langle\{a, e\}\{d\}\rangle$
sp_4	$\{g_2, g_3\}$	$\langle\{d\}\{a\}\{b\}\rangle$
sp_5	$\{g_2, g_3\}$	$\langle\{d\}\{a\}\{f\}\rangle$
sp_6	$\{g_2, g_3\}$	$\langle\{d\}\{b\}\{f\}\rangle$
sp_7	$\{g_1\}$	$\langle\{a, e\}\{c\}\{d\}\{a\}\rangle$
sp_8	$\{g_2\}$	$\langle\{d\}\{a, b, e\}\{f\}\{b, c, d\}\rangle$
sp_9	$\{g_3\}$	$\langle\{d\}\{a\}\{b\}\{f\}\rangle$

Table 3.3: The set of closed sequential patterns $\mathcal{CSP}(\mathcal{S}_{DB}, \sigma_s)$ with its tid list in the sequential dataset $\mathcal{S}_{DB}$ (see Figure 3.5.a) with respect to a minimal support threshold equal to 1.

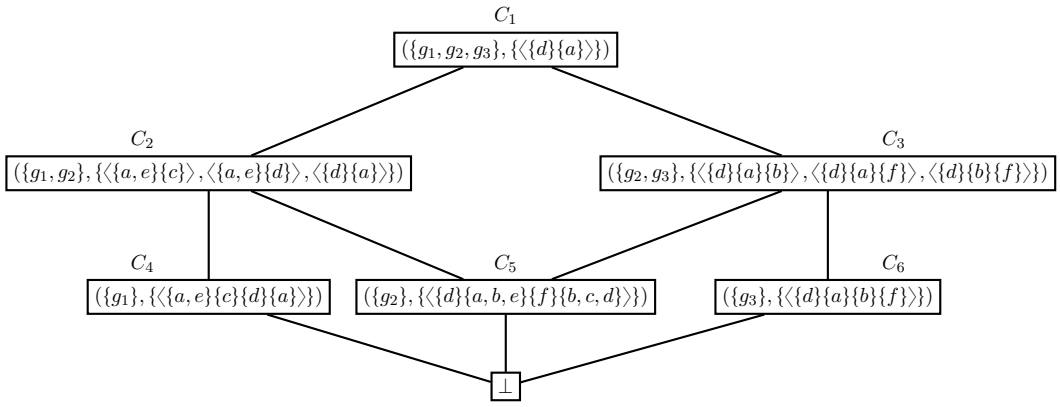


Figure 3.6: Concept lattice raised from ordered by applying the Algorithm 5 on the closed sequential pattern in Table 3.3

	sp_1	sp_2	sp_3	sp_4	sp_5	sp_6	sp_7	sp_8	sp_9
s_1	×	×	×				×		
s_2	×	×	×	×	×	×		×	
s_3	×			×	×	×			×

Table 3.4: Formal context

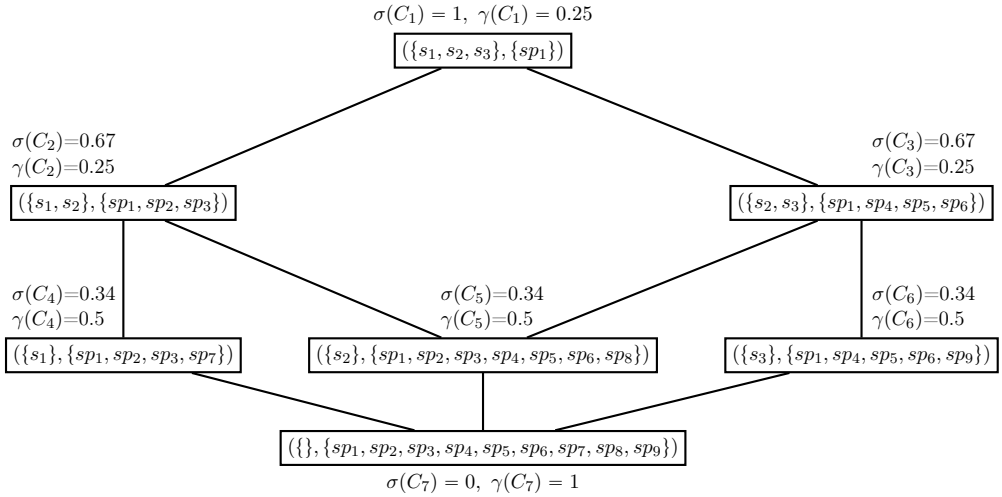


Figure 3.7: Concept lattice raised from formal context in Table 3.4 and by applying *Bordat* algorithm on the formal context in Table 3.4

Given a sequential patterns extracted from a sequential database, the formal context is built by taking the sequences in the database as objects, and the sequential patterns which are extracted for it as attributes. The sequence s is related to a sequential pattern sp if s supports of the pattern sp. Then, we use a standard FCA method like *Bordat* Bordat [1986] to build a concept lattice of sequences. For each concept in this lattice, its intent is the set of one or more sequential patterns while its extent is the set of all the sequences which are support all the sequential patterns in the intent.

In order to choose the most important concepts, we rely on stability and support as measures for pruning the concept lattice. Stability offers an alternative point of view on concepts compared to the well known metric of support based on frequency, which is used for building iceberg lattices Stumme [2002].

Choosing FCA as a method of classification and relying on stability and support as measures to filter concepts help us to find group of interesting sequential patterns and achieve a classification of sequences according to their patterns.

Example 31. *Table 3.4 shows a formal context* $\mathbb{K}$ *representing the binary relation between the sequences in Figure 3.5.a and its closed sequential patterns in Table 3.3. The cross indicates that the sequence supports the pattern. Figure 3.7 shows concept lattice raised from formal context in Table 3.4 and by applying* Bordat *algorithm.*

3.6 Multidimensional Sequential Pattern Structures

In this section, we will present a novel way of dealing with multidimensional sequences by mapping them to *pattern structures* Ganter and Kuznetsov [2001a].

3.6.1 Pattern Structures

FCA processes only the binary context, more complex data as sequences or graph can be directly processed as well. The power provided by the pattern structures allows our proposition to be directly instantiated with state-of-the-art FCA. The pattern structures were introduced in Ganter and Kuznetsov [2001b] as follows.

Definition 54. *A pattern structure is a triple $(G, (D, \sqcap), \delta)$, where G is a set of objects, $(D, \sqcap)$ is a complete meet-semilattice of descriptions and $\delta : G \rightarrow D$ maps an object to a description.*

Standard FCA can be presented in terms of a pattern structure where G is the set of objects, the semilattice of descriptions is $(\mathcal{P}(M), \sqcap)$ and a description is a set of attributes. In this case, $\sqcap$ operation corresponding to the intersection set operation. For example, If $x = \{a, b, c\}$ and $y = \{a, c, d\}$ where $x, y \in \mathcal{P}(M)$, then $x \sqcap y = x \cap y = \{a, c\}$. The mapping $\delta : G \rightarrow \mathcal{P}(M)$ is given by, $\delta(g) = \{m \in M \mid (g, m) \in I\}$, and returns the description for a given object as a set of attributes. In general, the lattice operation in the semilattice $(\sqcap)$ is called a *similarity operation* between two descriptions.

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

$$A^{\diamond} := \bigsqcap_{g \in A} \delta(g), \qquad \text{for } A \subseteq G$$

$$d^{\diamond} := \{g \in G \mid d \sqsubseteq \delta(g)\}, \qquad \text{for } d \in D$$

Galois connection provides a particular mapping between sets of objects and descriptions where $A^{\diamond}$ is the set of description which is common to all objects in A while $d^{\diamond}$ is the set of all objects whose description subsumes d.

More precisely, the partial order on D $(\sqsubseteq)$ is defined w.r.t. the similarity operation $\sqcap$ where:

$$c \sqsubseteq d \Leftrightarrow c \sqcap d = c$$

Definition 55. *A pattern concept of a pattern structure $(G, (D, \sqcap), \delta)$ is a pair (A, d) where $A \subseteq G$ and $d \in D$ such that $A^{\diamond} = d$ and $d^{\diamond} = A$, A is called the concept extent and d is called the concept intent.*

As, the stability and the support of the concepts only depend on the extent. Thus, they can be defined by the same procedure for both formal contexts and pattern structures.

3.6.2 Sequential Data

A sequence is constituted of elements from an alphabet. The alphabet can be either an itemset like classical sequence Srikant and Agrawal [1996], a graph Huan et al. [2004] or an elementary vector like the multidimensional sequence which is presented in *MMISP* (Chapter 2). We will generalize the definition of *subsequence/supersequence*. Suppose that the alphabets of a sequence have been ordered over a semilattice $(E, \sqcap)$ where the partial order, $\sqsubseteq$, on E is defined w.r.t the similarity operation $\sqcap$. In a multidimensional case, the $\sqcap$ operation between two elementary

$\mathcal{D}_{ex}=$

Patients	Trajectories
s_1	$\langle(h_1,\{a\}),(h_1,\{c,d\}),(h_1,\{a,b\}),(h_1,\{d\})\rangle$
s_2	$\langle(h_2,\{c,d\}),(h_3,\{b,d\}),(h_3,\{a,d\})\rangle$
s_3	$\langle(h_4,\{c,d\}),(h_4,\{b\}),(h_4,\{a\}),(h_4,\{a,d\})\rangle$

Table 3.5: An example of a database of patient trajectories.

vectors $e_1 = (e_{11}, ..., e_{1k})$ and $e_1 = (e_{21}, ..., e_{2k})$ corresponds to the similarity operation between i^{th} element in e_1 and e_2, i.e., $e_1 \sqcap e_2 = (e_{11} \sqcap e_{21}, ..., e_{1k} \sqcap e_{2k})$, while the $\sqsubseteq$ operation corresponds to the partial order between the elementary vectors. This kind of generalization allows one to process in a unified way all types of sequential data.

Definition 56. *(Subsequence/Supersequence Relation)*
*A sequence $s_1 = \langle s_{11} \cdots s_{1m}\rangle$ is a **subsequence** of $s_2 = \langle s_{21} \ldots s_{2n}\rangle$, denoted by $s_1 \leqslant s_2$, if there exist indices $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$ such that $s_{1j} \sqsubseteq s_{2i_j}$ for all $j = 1 \ldots m$ and $m \leqslant n$. s_2 is said to be a **supersequence** of s_1.*

We use the sequence database $\mathcal{D}_{ex}$ in Table 3.5 as a running example for a multidimensional sequence. It contains an example of medical trajectories with three patients. There are a set of medical procedures $MP = \{a, b, c, d\}$ a set of hospital names $H = \{h_1, h_2, h_3, h_4, cl, ch, \top_h\}$, where hospital names are hierarchically organized (by level of generality), h_1 and h_2 are central hospitals (ch) and h_3 and h_4 are clinics (cl), and $\top_h$ denotes the root of this hierarchy. The least common ancestor in this hierarchy is denoted as $h_i \sqcap h_j$, for any $h_i, h_j \in H$, i.e. $h_1 \sqcap h_2 = ch$. Every hospitalization is described with one hospital name and may contain several medical procedures. The medical procedure order in each hospitalization is not important in our case. For example, the first hospitalization $(h_2, \{c, d\})$ for the second patient (s_2) was in hospital h_2 and during this hospitalization patient underwent procedures c and d. the sequence $s_1 = \langle(ch, \{d\}), (cl, \{a\})\rangle$ is a subsequence of $s_2 = \langle(h_2, \{c, d\}), (h_3, \{b, d\}), (h_3, \{a, d\})\rangle$ as $(ch, \{d\}) \sqsubseteq (h_2, \{c, d\})$ and $(cl, \{a\}) \sqsubseteq (h_3, \{a, d\})$. A restriction is considered about the definition of subsequences (Definition 56) where only the order of consequent elements is taken into account. This restriction prompts us to define *substring/superstring* relation as follows:

Definition 57. *(Substring/Superstring Relation)*
*A sequence $s_1 = \langle s_{11} \cdots s_{1m}\rangle$ is a **substring** of $s_2 = \langle s_{21} \ldots s_{2n}\rangle$, denoted by $s_1 \leqslant s_2$, if there exist indices $i_j = i_{j-1} + 1$ such that $s_{1j} \sqsubseteq s_{2i_j}$ for all $j \in \{2 \ldots m\}$ and $m \leqslant n$. s_2 is said to be a **superstring** of s_1.*

For example, given a sequence $s_2 = \langle(h_2, \{c, d\}), (h_3, \{b, d\}), (h_3, \{a, d\})\rangle$ from Table 3.5, the sequence $\langle(cl, \{d\}), (cl, \{\})\rangle$ is a substring of s_2 while $\langle(ch, \{d\}), (cl, \{a\})\rangle$ does not consider as a substring of s_2. Table 3.6 shows an example of some substrings extracted from the sequences in Table 3.5.

3.6.3 Multidimensional Sequential Meet-semilattice

Based on the previous section, we will define the sequential pattern structure for representing and managing multidimensional sequences. Our definition is based on the pattern structures for graphs Kuznetsov [1999] where the authors presented the meet-semilattice operation $\sqcap$ respects subgraph isomorphism. Our sequential meet-semilattice respects the substring relation. Given an alphabet lattice $(E, \sqcap)$, $\mathcal{S}$ is the set of all sequences based on $(E, \sqcap)$. Based on Definition

	subsequence
sp_1	$\langle(ch, \{c, d\}), (h_1, \{b\}), (\top_h, \{h\})\rangle$
sp_2	$\langle(ch, \{\}), (\top_h, \{d\}), (\top_h, \{a\})\rangle$
sp_5	$\langle(\top_h, \{a\})\rangle$
sp_7	$\langle(cl, \{d\}), (cl, \{\})\rangle$
sp_9	$\langle(ch, \{c, d\})\rangle$
sp_{11}	$\langle(\top_h, \{c, d\}), (\top_h, \{b\})\rangle$

	subsequence
sp_2	$\langle(ch, \{c, d\}), (\top_h, \{b\}), (\top, \{d\})\rangle$
sp_4	$\langle(\top_h, \{c, d\}), (\top, \{b\})\rangle$
sp_6	$\langle(\top_h, \{c, d\}), (cl, \{b\}), (cl, \{a\})\rangle$
sp_8	$\langle(cl, \{\}), (cl, \{a, d\})\rangle$
sp_{10}	$\langle(cl, \{b\}), (cl, \{a\})\rangle$
sp_{12}	$\langle(\top_h, \{a\}), (\top_h, \{d\})\rangle$

Table 3.6: Substring of patients sequences in Table 3.5

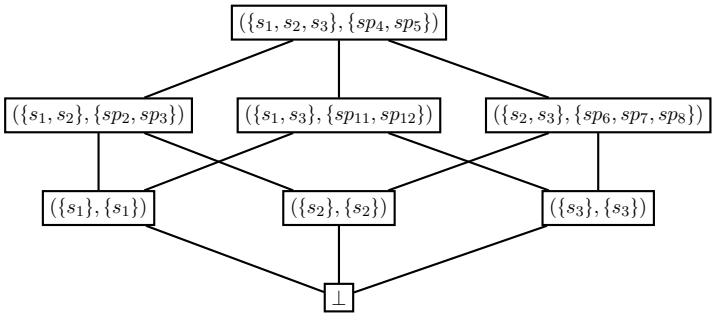


Figure 3.8: The concept lattice for the pattern structure given by Table 3.5. Concept intents reference to sequences in Tables 3.5 and 3.6

57, the set $\mathcal{S}$ is a partially ordered. $(D, \sqcap)$ is a semilattice on the partial order set $\mathcal{S}$, where $D \subseteq \mathcal{P}(\mathcal{S})$, such that, if a set of sequences d in the semilattice D contains a sequence s, then all the substrings of s should be included into the set d. More formally, $\forall s \in d; \nexists s' \leqslant s \text{ and } s' \notin d$. The similarity operation $\sqcap$ between two sets of sequences in D is the *set intersection* operation. Given two sets of sequence d_1 and d_2 in the semilattice D, the *set intersection* operation ensures that, if a sequence s belong to $d_1 \sqcap d_2$, then any substrings of s belongs also to $d_1 \sqcap d_2$, thus $(d_1 \sqcap d_2) \in D$. Finally, we can say that $(D, \sqcap)$ is a valid semilattice as the *set intersection* operation is idempotent, commutative and associative.

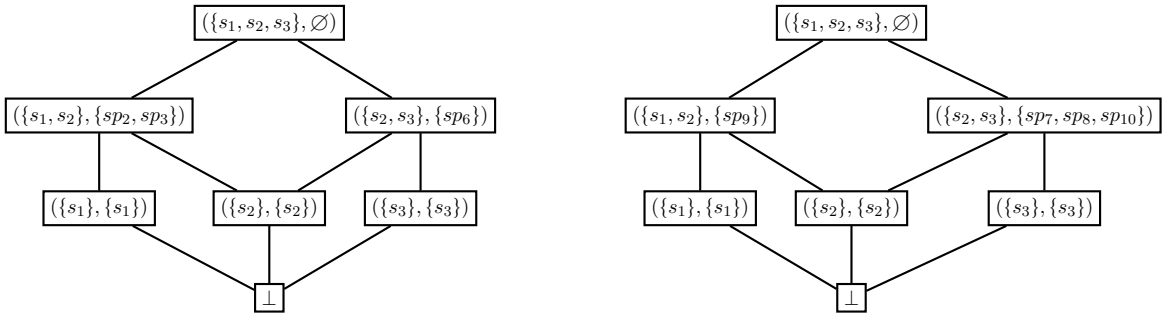
The set of all possible substring for a given sequence can be rather large. It is more efficient and representable to keep a pattern $d \in D$ as a set of all maximal substrings, denoted by d_{max} which is defined as follows:

$$d_{max} = \{s_1 \in d; \nexists s_2 \in d; s_2 \leqslant s_1\}$$

Every pattern will be given only by the set of all maximal common substrings. For example, $\{s_2\} \sqcap \{s_3\} = \{sp_6, sp_7, sp_8\}$ (see Tables 3.5 and 3.6), i.e. $\{sp_6, sp_7, sp_8\}$ is the set of all maximal common substrings of two sequences s_2 and s_3, correspondingly $\{sp_6, sp_7, sp_8\} \sqcap \{s_1\} = \{ss_4, ss_5\}$.

Given two sets of sequences $s = \{s_1, ...s_n\}$ and $t = \{t_1, ..., t_m\}$, the similarity between these sets, $s \sqcap t$, is calculated according to the maximal sequences among all common substring for any pair of s_i and t_j. To find all common substrings of two sequences s_i and t_j, we first align s_i and t_j in all possible ways. For each alignment of s_i and t_j we compute the resulting intersection. Finally, we keep only the maximal intersected substrings.

Let us consider two possible alignments of $s_1 = \langle\{a\}, \{c, d\}, \{a, b\}, \{d\}\rangle$ and $t_1 = \langle\{c, d\}, \{b, d\}, \{a, d\}\rangle$:



(a) MLP projection $l = 3$ (b) Projection by removing $\top_h$ value

Figure 3.9: The projected concept lattices for the pattern structure given by Table 3.5. Concept intents reference to sequences in Tables 3.5 and 3.6.

$$
\begin{array}{rl}
s_1 = & \langle \{a\}, \{c,d\}, \{a,b\}, \ \{d\} \ \rangle \\
t_1 = & \langle \{c,d\}, \{b,d\}, \{a,d\} \rangle \\
y_1 = & \langle \{c,d\}, \ \{b\}, \ \ \{d\} \ \rangle
\end{array}
\qquad
\begin{array}{rl}
s_1 = & \langle \{a\}, \{c; d\}, \{a,b\}, \ \{d\} \ \rangle \\
t_1 = & \langle \{c,d\}, \{b,d\}, \{a,d\} \rangle \\
y_2 = & \langle \ \varnothing \ , \ \{d\} \ \rangle
\end{array}
$$

The right intersection $y_2 = \langle \{d\} \rangle$ is not retained, as it is not maximal, while the left intersection $y_1 = \langle \{c,d\}, \{b\}, \{d\} \rangle$ is kept.

Example 32. *The sequential pattern structure for our example (see Table 3.5) is* $(G, (D, \sqcap), \delta)$*, where* $G = \{s_1, s_2, s_3\}$*,* $(D, \sqcap)$ *is the semilattice of sequential descriptions, and* δ *is the mapping associating an object in* G *to a description in* D *shown in Table 3.5. Figure 3.8 shows the resulting lattice of sequential pattern concepts for this particular pattern structure* $(G, (D, \sqcap), \delta)$*.*

FCA algorithm can be adapted to process pattern structures instead of standard FCA contexts. We adapted *AddIntent* algorithm Van Der Merwe et al. [2004] to build a lattice of pattern concepts . To adapt the algorithm to our needs, every set intersection operation on attributes should be substituted with semilattice operation $\sqcap$ on corresponding patterns, while every subset checking operation should be substituted with semilattice order checking $\sqsubseteq$.

3.7 Projections of Sequential Pattern Structures

Due to huge number of concepts in the concept lattice, complicated descriptions of objects and similarity operation, pattern structures may give rise to many computational challenges. Given a pattern structure, the produced concept lattice may generate many useless results which an expert may not be interested in. Now the question arises: *Is there any way to decrease computation time by calculating only those patterns which are interesting for the experts?*. In order to do so projections on pattern structures may help in simplifying computation process by allowing the user to reduce the descriptions. Projections can be considered as filters on patterns w.r.t. some mathematical properties Ganter and Kuznetsov [2001a].

A possible projection for sequential pattern structures comes from the following observation. In many cases it may be more interesting to analyze long subsequences rather than small ones. We call these projections *Minimal Length Projection* (MLP) and they depend on the minimal allowed length l for the sequences in a pattern. To project a pattern structure w.r.t. MLP, a pattern should be substituted with the pattern where any sequence of length less then l is removed. For example, if we set the minimal length threshold to 3, then there is only one maximal common

subsequence sp_6 in Table 3.6 between the twos sequences s_2 and s_3 in Table 3.5, while sp_7 and sp_8 are too short to be considered. Figure 3.9.a shows the lattice corresponding the projected pattern structure (Table 3.5) with sequential patterns of length more or equal 3.

Another important type of projections is connected to a variation of the lattice alphabet $(E, \sqcap)$. The simplest variation of the alphabet is to ignore of certain fields in the elements. For example, if a hospitalization is described by a hospital name and a set of procedures, then either hospital or procedures can be ignored in similarity computation. For that, in any element a hospital can be substituted by $\top_h$ which is the most general element of the taxonomy of hospitals.

Another variation of the alphabet, is to require that some field(s) should not be empty. For example, we want to find patterns with non-empty set of procedures, or we want to have information about hospital (the element $\top_h$ of hospital taxonomy is not allowed in an element of a sequence). Such variations are easy to realize within our approach. For this, when computing the similarity operation between elements of the alphabet, one should check if the result contains empty fields and, if yes, should skip this alphabet,. For example, an expert is interested in finding sequential patterns describing how a patient changes hospitals, without interest in procedures. Thus, any element of the alphabet lattice containing a non-empty set of procedures is projected to the corresponding element with the same hospital and an empty set of procedures. Similarly, an expert is interested in finding sequential patterns containing information about the hospital in every hospitalization, and the corresponding procedures, i.e. hospital field in the patterns cannot be $\top_h$, e.g., sp_5 is an invalid pattern, while sp_6 is a valid pattern in Table 3.8. Figure 3.9.b shows the lattice corresponding to the projected pattern structure (Table 3.5) by changing the alphabet semilattice.

3.8 Conclusion

In this chapter, we addressed the problem of analysing and mining sequential data with the Formal Concept Analysis. We present a way of regrouping the sequential patterns extracted from sequential database by using a standard FCA. We show some interesting properties of concept lattices and stability index to classify and select interesting patterns. Then, we propose a novel way of dealing with sequences by mapping them to pattern structures. The genericity power provided by the pattern structures allows our approach to be directly instantiated with state-of-the-art FCA algorithms. Then, we introduce and discuss the notion of *"projections"* for sequential pattern structures. These mathematical objects significantly decrease the number of patterns, while preserving the most interesting ones for an expert. Projections are easily built to answer questions that an expert may have.

For future work, we are planning to more deeply investigate projections, their potentialities w.r.t. the types of patterns. It can be interesting to introduce and evaluate the stability measure directly on sequences. Another research direction is mining of association rules or building a Horn approximation Balcázar and Garriga [2007] from the stable part of the pattern lattice or stable sequences.

Chapter 4

On Measuring Similarity for Sequences of Itemsets

Contents

4.1 Introduction 73
4.2 Related Work 75
4.2.1 Edit distance 75
4.2.2 Dynamic Time Warping (DTW) 76
4.2.3 Similarity Measure for Sequential Patterns (S^2MP) 77
4.2.4 Longest Common Subsequence (LCS) 77
4.2.5 All Common Subsequences (ACS) 78
4.3 Preliminaries 78
4.4 Longest And All Common Subsequences: A Comparison 81
4.5 Counting All Distinct Subsequences 81
4.6 Counting All Common Subsequences 84
4.6.1 Dynamic Programming 87
4.7 Complexity and Linial-Nisan Approximation Results 88
4.7.1 Complexity 88
4.7.2 Linial-Nisan Approximation 90
4.8 Implementation and Experimental Validation 92
4.8.1 Clustering Handwritten Assamese Symbols 93
4.8.2 Linial-Nisan approximation 96
4.8.3 Studying Scalability on Synthetic Data Sets 97
4.9 Conclusion and Perspective 98

4.1 Introduction

Sequential data is widely present and used in many applications such as matching of time series in databases Faloutsos et al. [1994], DNA or amino-acids protein sequence analysis Sander and Schneider [1991], Chothia and Gerstein [1997], web log analysis Yang and Zhang [2003], and music sequences matching Serrà et al. [2012]. Consequently, analyzing sequential data has become an important data mining and machine learning task with a special focus on the examination

of pairwise relationships between sequences. For example, some clustering and kernel-based learning methods depend on computing distances or similarity scores between sequences Leslie et al. [2002], Xiong et al. [2011].

However, in many applications, similarity measures on sequential data remain limited to *simple sequences*, which are ordered lists of items (i.e., symbols) Levenshtein [1966], Herranz et al. [2011], Keogh [2002], Wang and Lin [2007]. By contrast, in modern life sciences Wodak and Janin [2002], sequential data sets are often represented as ordered lists of symbol sets (i.e., *itemsets*). This special feature is in itself a challenge and implies to carefully take into account complex combinatorial aspects to compute similarities between sequences.

In this chapter, we focus on the notion of common subsequences[6] as a means to define a distance or similarity score between a pair of sequences composed of a list of itemsets. The assumption that common subsequences can characterize similarity is not new. For instance, a very well known state-of-the-art algorithm, *longest common subsequence* Hirschberg [1975], uses the length of the longest common subsequence as a similarity measure between two sequences. However, as clearly stated by Wang and Lin [2007] for simple sequences:*"This measure [...] ignores information contained in the second, third, ..., longest subsequences"* . Additionally, this measure behaves erratically when the sequences contain itemsets. We justify this claim by considering three sequences $U = \langle\{c\}\{b\}\{a,b\}\{a,c\}\rangle$, $V = \langle\{b\}\{c\}\{a,b\}\{a,c\}\rangle$ and $W = \langle\{b,d\}\{a,b\}\{a,c\}\{d\}\rangle$. The longest common subsequence, denoted by LCS, between sequences U and V is $LCS(U,V) = \langle\{b\}\{a,b\}\{a,c\}\rangle$, and between U and W is $LCS(U,W) = \langle\{b\}\{a,b\}\{a,c\}\rangle$. This similarity measure is usually defined as $sim_{LCS}(S,T) = \frac{|LCS(S,T)|}{max(|S|,|T|)}$ and thus one may conclude that because $sim_{LCS}(U,V) = sim_{LCS}\ (U,W) = \frac{3}{4}$, then the sequence U is equidistant from sequence V and W. Clearly this is a wrong result as U is almost the same sequence as V, but with a slight inversion of the two first itemsets. *How can one maximize the information used to compute a similarity measure between two sequences?*

Along with Wang and Lin [2007], we strongly believe that the *number of common subsequences* (and not only the length of the longest one) between two sequences is appealing in order to answer the previous question. We illustrate this intuition with the three previously considered sequences U, V and W. Let $ACS(S,T)$ be the cardinality of the set that contains *all common subsequences* between S and T. An example of a common subsequence between U and V is $\langle\{c\}\{a,b\}\rangle$. Notice, on the other hand, that $\langle\{c\}\{a,b\}\rangle$ is not a common subsequence between U and W. $ACS(U,V) = 40$, $ACS(U,W) = 26$ and $ACS(V,W) = 26$. Based on this computation, it is trivial to conclude that sequences U and V share stronger affinity than with W (a result that was not detected by the longest common subsequence measure).

To date, there *does not exist* any approach that computes efficiently ACS and use it as a basis for a similarity measure for complex sequences. Accordingly, in this work, the main contributions are summarized as follows:

Algorithmic and approximability results. We discuss and present a dynamic programming algorithm for counting all common subsequences (ACS) between two given sequences. This dynamic programming algorithm allows us to define in a simple and intuitive manner our similarity measure which is the ratio between the number of common subsequences from two sequences S and T divided by the maximal number of distinct subsequences. To cope with large data sets containing long input sequences, we present an *approximation technique* to compute efficiently ACS. The approach relies on approximating the size of a union of a family of sets in terms of the intersections of all subfamilies (i.e., *inclusion-exclusion principle*) based on the direct application

[6] A subsequence is a sequence that can be derived from another sequence by deleting items without changing the order of itemsets. The notion of subsequence will be further developed in Section 4.3.

of a result from Linial and Nisan [1990].

Experiments and Evaluations. The results reported in this work are a useful contribution with direct practical applications to different discriminative approaches, and in particular kernel methods, because new complex sequence kernels can be devised based on the theoretical results provided in this work. Moreover, the method is quite general in that it can be used (with slight modifications) for a broad spectrum of sequence-based classification or clustering problems. We report an extensive empirical study on synthetic datasets and qualitative experiments with online dataset about handwritten Assamese characters. We give empirical evidence showing that the proposed approximation method works well in practice. The different versions of the software source codes, data used for experiments and interactive data visualizations are publicly available from `http://www.loria.fr/~eegho/acs/`.

The rest of the chapter is organized as follows. Section 4.2 reviews the related work. Section 4.3 briefly reviews the preliminaries needed in our development. Section 4.4 highlights the differences between our similarity measure and the well-known *"longest common subsequence"* along with a discussion on the motivations and the practical impact of our measure. Section 4.5 and 4.6 introduces our new combinatorial results. Section 4.7 presents a complete study of the complexity of the problem and an efficient approximation technique.

4.2 Related Work

There are a variety of methods used in different applications for finding the similarity between two or more sequences. In this section, we give a brief introduction to existing sequence similarity measures.

4.2.1 Edit distance

Edit distance is a string metric for measuring the difference between two sequences of items not itemsets proposed by Levenshtein [1966]. *Edit distance* is often used to refer specifically to *Levenshtein distance*. It focuses on the lexical level and consider the insertion and deletion of an item, the substitution of two items, and the transposition of two adjacent items as a unit of difference between two string sequences Levenshtein [1966].

Using the dynamic programming Bellman et al. [1959], the edit distance between two sequences S and T is computed as follows Wagner and Fischer [1974]

$$sim_{Edit}(S^i, T^j) = \begin{cases} max(i,j) & if\ min(i,j) = 0 \\ min \begin{cases} sim_{Edit}(S^{i-1}, T^j) + 1 \\ sim_{Edit}(S^i, T^{j-1}) + 1 \\ sim_{Edit}(S^{i-1}, T^{j-1}) + 1_{S[i] \neq T[j]} \end{cases} & \text{otherwise.} \end{cases}$$

Where S^i denotes the i-prefix of sequence S, $S[i]$ is the ith item of sequence S and $1_{S[i] \neq T[j]}$ is the indicator function equal to 0 when $S[i] = T[j]$ and 1 otherwise. Then, Edit distance between two sequences S and T is computed as $sim_{Edit}(S,T) = sim_{Edit}(S^{|S|}, T^{|T|})$.

Kum et al. [2003] used *Edit distance* as a measure to cluster sequences of itemsets in a sequential database. The *Edit distance* between two sequences of itemests S and T can be computed by dynamic programming using the following recurrence relation.

$$sim_{Edit}(S^i, T^j) = \begin{cases} max(i,j) & if\ min(i,j) = 0 \\ min \begin{cases} sim_{Edit}(S^{i-1}, T^j) + 1 \\ sim_{Edit}(S^i, T^{j-1}) + 1 \\ sim_{Edit}(S^{i-1}, T^{j-1}) + REPL(S[i], T[j]) \end{cases} & \text{otherwise.} \end{cases}$$

Where,

$$\begin{aligned} REPL(X,Y) &= \frac{|(X \backslash Y) \cup (Y \backslash X)|}{|X| + |Y|} \\ &= \frac{|X| + |Y| - 2 \cdot |X \cap Y|}{|X| + |Y|} \end{aligned}$$

The Edit distance has found a large variety of applications in many scenarios and has achieved very good results Ristad and Yianilos [1998]. Mitton [2009] proposed a combination of an edit distance and letter match method. Considering the length of sequences, Li and Liu [2007] proposed a normalized edit distance metric. Bilenko and Mooney [2003] presented a learnable edit distance metric that is based on a support vector machine (SVM). Many other approaches are built on this result but with notable differences like weighting the symbols and the edit operations Herranz et al. [2011], or using stochastic processes Oncina and Sebban [2006].

4.2.2 Dynamic Time Warping (DTW)

Dynamic time warping (DTW) is a well-known technique to find an optimal alignment between two time series Berndt and Clifford [1994]. Intuitively, the sequences are warped in a nonlinear fashion to match each other.

Suppose we have two time series S and T, of length n and m respectively. To align two time series using DTW, Berndt and Clifford [1994] construct a n-by-m matrix. Each matrix element (i,j) corresponds to the Euclidean distance Deza and Deza [2009] $d(S[i], T[j])$ between the two points S[i] and T[j]. A warping path W, is a contiguous set of matrix elements that defines a mapping between S and T. The kth element of W is defined as $w_k = (i,j)_k$ so we have:

$$W = w_1, w_2, ..., w_K \ where\ max(m,n) \leqslant K \leqslant m + n - 1$$

The warping path is typically subject to several constraints:

Boundary conditions: $w_1 = (1,1)$ and $w_K = (m,n)$, simply stated, this requires the warping path to start and finish in diagonally opposite corner cells of the matrix.

Continuity: Given $w_k = (a,b)$ then $w_{k-1} = (a^{'}, b^{'})$ where $a - a^{'} \leqslant 1$ and $b - b^{'} \leqslant 1$. This restricts the allowable steps in the warping path to adjacent cells.

Monotonicity: Given $w_k = (a,b)$ then $w_{k-1} = (a^{'}, b^{'})$ where $a - a^{'} \geqslant 0$ and $b - b^{'} \geqslant 0$. This forces the points in W to be monotonically spaced in time.

There are exponentially many warping paths that satisfy the above conditions, however DTW is interested only in the path which minimizes the warping cost:

$$sim_{DTW}(S,T) = min\big(\frac{\sum_{k=1}^{K} w_k}{K}\big)$$

This path can be found very efficiently using dynamic programming as follows:

$$\gamma(i,j) = d(S[i],T[j]) + min\{\gamma(i-1,j-1),\gamma(i-1,j),\gamma(i,j-1)\}$$

Where $d(S[i],T[j])$ is a Euclidean distance Deza and Deza [2009] between two point $S[i]$ and $T[i]$, $\gamma(0,0) = 0$, $\gamma(i,0) = +\infty$ $(\forall i; 1 \leqslant i \leqslant |S|)$ and $\gamma(0,j) = +\infty$ $(\forall j; 1 \leqslant j \leqslant |T|)$.

Then,

$$sim_{DTW}(S,T) = \gamma(|S|,|T|)$$

DTW had a huge impact and has been used to compare multiple patterns in automatic speech recognition to cope with different speaking speeds Muzaffar et al. [2005]. In addition to speech recognition, dynamic time warping is still widely used in various fields. In bioinformatics, Aach and Church [2001] successfully applied DTW to RNA expression data. In chemical engineering, it has been used for the synchronization and monitoring of batch processes in polymerization Gollmer and Posten [1995]. DTW has been successfully used to align biometric data, such as gait Gavrila and Davis [1995], signatures Munich and Perona [1999] and even fingerprints Kovacs-Vajna [2000]. Many researchers, including Caiani et al. [1998], have demonstrated the utility of DTW for ECG pattern matching. Rath and Manmatha [2003] have applied DTW to the problem of indexing repositories of handwritten historical documents.

4.2.3 Similarity Measure for Sequential Patterns (S^2MP)

Saneifar et al. [2008] defined a similarity measure, S^2MP, adapted to sequential patterns taking into account all the characteristics of sequential patterns and in particular their semantics. The degree of similarity is the result of the aggregation of two scores.

1. The *mapping score* which measures the resemblance of two sequences based on the links that can be established between itemset, *aveWeightScore*.

2. The *order score* which measures the resemblance of the two sequences based on the order and positions of itemsets in sequences,*orderScore*.

The combination of two independent scores allows a modular measurement. It is therefore adaptable and parametrizable depending on the context as follows:

$$sim_{S^2MP}(S,T) = \frac{(orderScore(S,T) \times Co_1) + (aveWeightScore(S,T) \times Co_2)}{Co_1 + Co_2}$$

S^2MP measure is not symmetrical where $sim_{S^2MP}(S,T) \neq sim_{S^2MP}(T,S)$.

4.2.4 Longest Common Subsequence (LCS)

Karlton Sequeira [2002] and Vlachos et al. [2003] followed a radically different approach by developing longest common subsequences approaches for the comparison and similarity measure. As a computer science problem, LCS is concerned with the search for an efficient method of finding the longest common subsequence of two or more sequences Wang et al. [2010]. Using a dynamic programming algorithms, LCS calculates in polynomial time as follows Bergroth et al. [2000], Hirschberg [1977].

Suppose we have two sequences of items S and T, of length n and m respectively. Then:

$$sim_{LCS}(S^i, T^j) = \begin{cases} 0 & if i = 0 \text{ or } j = 0, \\ 1 + sim_{LCS}(S^{i-1}, T^{j-1}) & if\ S[i] = T[j], \\ max\{sim_{LCS}(S^i, T^{j-1}), sim_{LCS}(S^{i-1}, T^j)\} & if S[i] \neq T[j] \end{cases}$$

Then, the longest common subsequence between two sequences S and T is computed as $sim_{LCS}(S,T) = sim_{LCS}(S^{|S|}, T^{|T|})$.

4.2.5 All Common Subsequences (ACS)

However, the common information shared between two sequences is more than the longest common subsequence. In fact counting all possible common information units between sequences provides a good idea about the similarity relationship between the sequences and their overall *complexity*. In addition, the common subsequences problem is related to the problem of counting the number of all distinct common subsequences between two sequences of items.

Wang and Lin [2007] studied the usage of the count of all common subsequences (ACS) as a similarity measure between two sequences of items. Elzinga et al. [2008] followed the same intuition and proposed a dynamic programming algorithm to count distinct common subsequences between two sequences of items as follows.

Proposition 3. *Let S, T be two sequences of items of length n and m respectively. For each $X \in \mathcal{I}$, let $\ell(S,X) = max\{i;\ S[i] = X\}$ with $\ell(S,X) = 0$ if $\langle\{X\}\rangle \nsubseteq S$. For brevity, $\ell_S = \ell(S^{m-1}, S[m])$ and $\ell_T = \ell(T, S[m])$. Then, all common subsequences between S and T, denoted as $sim_{ACS}(S,T)$, is computed as follows:*

$$sim_{ACS}(S^i, T^j) = \begin{cases} sim_{ACS}(S^{i-1}, T^j) & if \langle\{S[i]\}\rangle \nsubseteq T^j, \\ sim_{ACS}(S^{i-1}, T^j) + sim_{ACS}(S^{i-1}, T^{\ell_T - 1}) & if \langle\{S[i]\}\rangle \subseteq T^j \text{ and } \langle\{S[i]\}\rangle \nsubseteq S^{i-1} \\ sim_{ACS}(S^{i-1}, T^j) + sim_{ACS}(S^{i-1}, T^{\ell_T - 1}) & \\ \qquad - sim_{ACS}(S^{\ell_S - 1}, T^{\ell_T - 1}) & if \langle\{S[i]\}\rangle \subseteq T^j \text{ and } \langle\{S[i]\}\rangle \subseteq S^{i-1} \end{cases}$$

In this chapter, we extend and generalize the previous works of Wang and Lin [2007], Elzinga et al. [2008] for complex structures such as sequences of itemsets.

4.3 Preliminaries

Definition 58 (Sequence)**.** *Let $\mathcal{I}$ be a finite set of* items*. An* itemset *X is a non-empty subset of $\mathcal{I}$. A* sequence *S over $\mathcal{I}$ is an ordered list $\langle X_1 \cdots X_n\rangle$, where X_i ($1 \leqslant i \leqslant n$, $n \in \mathbb{N}$) is an itemset. n is called the size of the sequence S, denoted by $|S| = n$. The length, denoted by $\ell(S)$, is the total number of items occurring in the sequence, i.e., $\ell(S) = \sum_{i=1}^{n} |X_i|$. S^l denotes the l-prefix $\langle X_1 \ldots X_l\rangle$ of sequence S with $1 \leqslant l \leqslant n$. The j-th itemset X_j of sequence S is denoted $S[j]$ with $1 \leqslant j \leqslant n$.*

Definition 59 (Subsequence)**.** *A sequence $T = \langle Y_1 \cdots Y_m\rangle$ is a **subsequence** of $S = \langle X_1 \ldots X_n\rangle$, denoted by $T \preceq S$, if there exist indices $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$ such that $Y_j \subseteq X_{i_j}$ for all $j = 1 \ldots m$ and $m \leqslant n$. S is said to be a **supersequence** of T. $\varphi(S)$ denotes the **set of all subsequences** of a given sequence S and $\phi(S) = |\varphi(S)|$. For two sequences S and T, $\varphi(S,T)$ denotes the set of **all common subsequences** between two sequences S and T: $\varphi(S,T) = \varphi(S) \cap \varphi(T)$ and $\phi(S,T) = |\varphi(S,T)|$.*

$\mathcal{D}_{ex} =$

S_1	$\langle\{a\}\{a,b\}\{e\}\{c,d\}\{b,d\}\rangle$
S_2	$\langle\{a\}\{b,c,d\}\{a,d\}\rangle$
S_3	$\langle\{a\}\{b,d\}\{c\}\{a,d\}\rangle$
S_4	$\langle\{a\}\{a,b,d\}\{a,b,c\}\{b,d\}\rangle$

Table 4.1: The sequence database used as a running example

We now define the following similarity measure between two sequences of itemsets S and T.

Definition 60 (Sequence Similarity)**.** *The similarity between two sequences S and T, denoted $sim_{ACS}(S,T)$ is defined as the number of common subsequences divided by the maximal number of subsequences of S and T; that is:*

$$sim_{ACS}(S,T) = \frac{\phi(S,T)}{\max\{\phi(S),\phi(T)\}}$$

The similarity measure satisfies the following conditions:

1. Non-negativity, $sim_{ACS}(S,T) \geqslant 0$;
2. Identity of indiscernibles, $sim_{ACS}(S,T) = 1$ if and only if $S = T$;
3. Symmetry, $sim_{ACS}(S,T) = sim_{ACS}(T,S)$.

However, sim_{ACS} does not respect the *triangular inequality* condition.

Definition 61 (Concatenation)**.** *Let $S = \langle X_1 \cdots X_n\rangle$ be a sequence, and Y be an itemset. The **concatenation of the itemset Y with the sequence** S, denoted $S \circ Y$, is the sequence $\langle X_1 \cdots X_n\ Y\rangle$.*

As usual, the powerset of an itemset Y will be denoted by $\mathcal{P}(Y)$, and $\mathcal{P}_{\geqslant 1}(Y)$ denotes all nonempty subsets of Y; that is, $\mathcal{P}_{\geqslant 1}(Y) = \mathcal{P}(Y)\backslash\{\varnothing\}$.

Example 33.*: We use the sequence database $\mathcal{D}_{ex}$ in Table 4.1 as a running example. It contains 4 data sequences over the set of items $\mathcal{I} = \{a,b,c,d,e\}$. Sequence $\langle\{a\}\{b\}\{c,d\}\rangle$ is a subsequence of $S_1 = \langle\{a\}\{a,b\}\{e\}\{c,d\}\{b,d\}\rangle$. $\ell(S_1) = 8$ and $|S_1| = 5$. The 3-prefix of S_1, denoted S_1^3, is $\langle\{a\}\{a,b\}\{e\}\rangle$ and $S_1[2]$, the second itemset in sequence S_1, is $\{a,b\}$. The set of all subsequences of S_4^2 is*

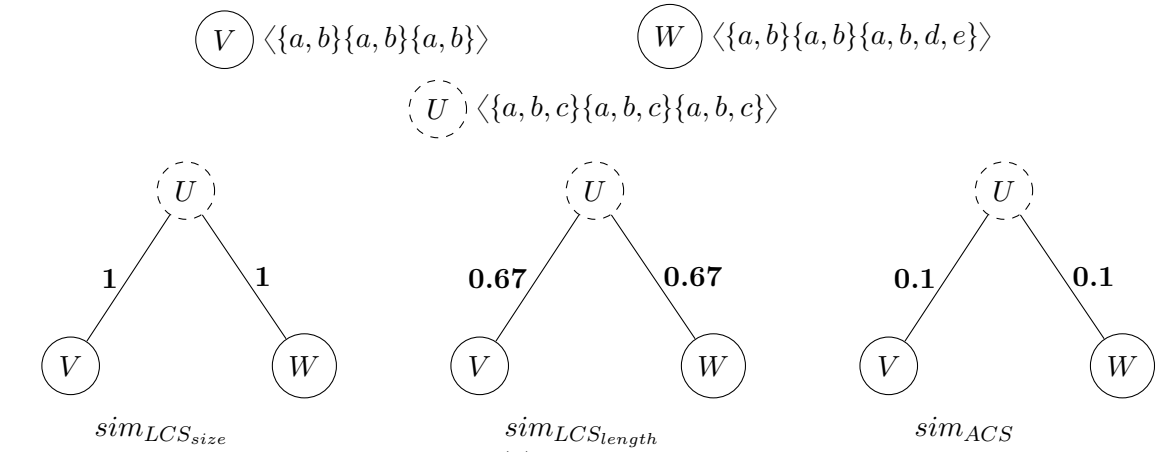
$$\begin{aligned}\varphi(S_4^2) = \{&\langle\rangle, \langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{d\}\rangle, \langle\{a,b\}\rangle, \langle\{a,d\}\rangle, \langle\{b,d\}\rangle, \langle\{a,b,d\}\rangle, \langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle,\\ &\langle\{a\}\{d\}\rangle, \langle\{a\}\{a,b\}\rangle, \langle\{a\}\{a,d\}\rangle, \langle\{a\}\{b,d\}\rangle, \langle\{a\}\{a,b,d\}\rangle\}\end{aligned}$$

Hence, $\phi(S_4^2) = 15$. The concatenation of the sequence S_4^2 with the itemset $\{a,b,c\}$, denoted as $S_4^2 \circ \{a,b,c\}$, is the sequence $\langle\{a\}\{a,b,d\}\{a,b,c\}\rangle$. In addition,the set of all common subsequences of S_1^4 and S_2^3 is

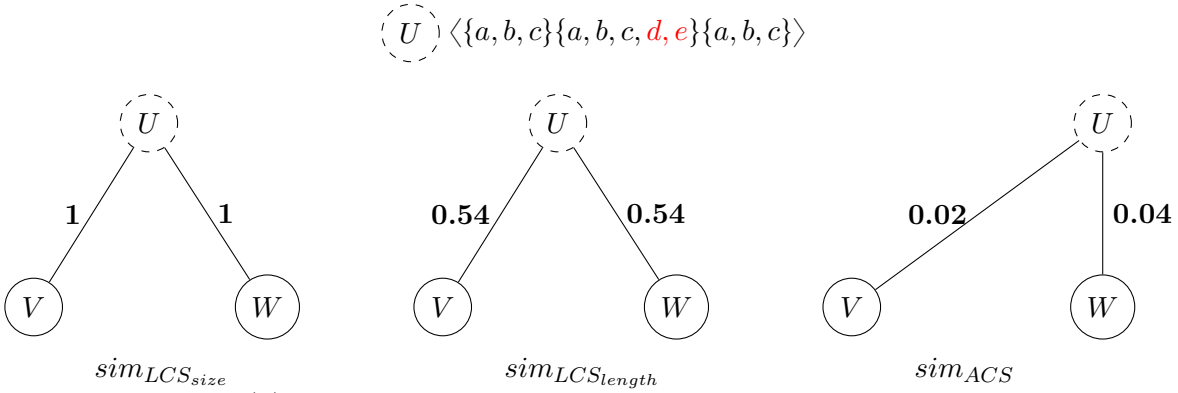
$$\begin{aligned}\varphi(S_1^4, S_2^3) = \{&\langle\rangle, \langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{d\}\rangle, \langle\{c\}\rangle, \langle\{c,d\}\rangle, \langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle, \langle\{a\}\{d\}\rangle,\\ &\langle\{a\}\{c,d\}\rangle, \langle\{b\}\{d\}\rangle, \langle\{a\}\{b\}\{d\}\rangle\}\end{aligned}$$

The similarity between S_1^4 and S_2^3 is

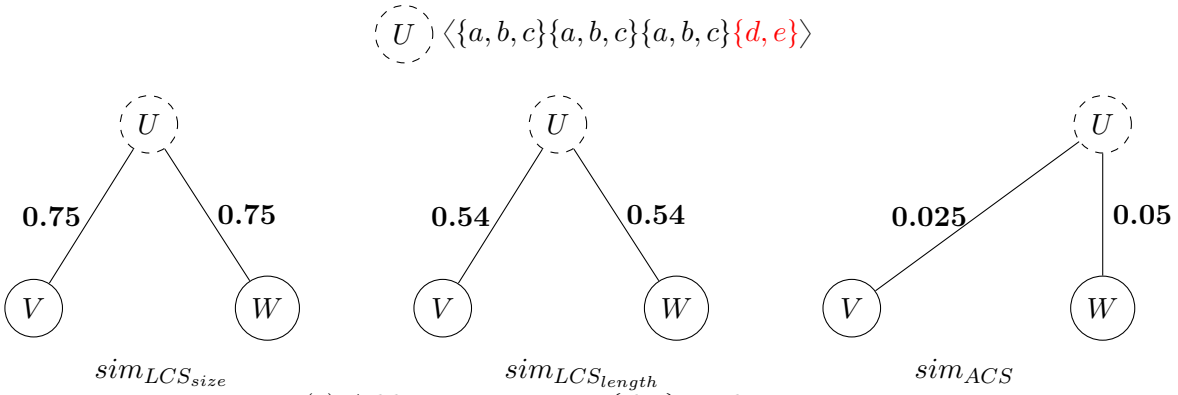
$$sim_{ACS}(S_1^4, S_2^3) = \frac{\phi(S_1^4, S_2^3)}{\max\{\phi(S_1^4), \phi(S_2^3)\}} = \frac{13}{\max\{56,61\}} = \frac{13}{61} = 0.21$$



(a) Original Case



(b) Adding new items d and e to the second itemset of U



(c) Adding new itemset $\{d, e\}$ to the sequence U

Figure 4.1: Sensitivity of different sequence similarity measures

4.4 Longest And All Common Subsequences: A Comparison

In this section, we review the well-known similarity measure *longest common subsequence* and our novel similarity based on *all common subsequences*. We briefly compare and contrast the different advantages of the proposed measure.

Using *longest common subsequence* for computing the similarity between two *sequences of itemsets* can lead to *ambiguous* results. There are two ways for measuring the longest common sequence in this case: either by using the *size* or the *length* of the sequence. If the size is used, the similarity between two sequences is defined as: $sim_{LCS_{size}}(S,T) = \frac{|LCS(S,T)|}{max\{|S|,|T|\}}$. If the sequence length is preferred, the similarity between two sequences becomes: $sim_{LCS_{length}}(S,T) = \frac{\ell(LCS(S,T))}{max\{\ell(S),\ell(T)\}}$.

Example 34. *Consider sequences* $S = \langle\{a,b\}\{a,c\}\{a,f\}\rangle$ *and* $T = \langle\{a,b,f\}\{c\}\rangle$. *The longest common subsequence between* S *and* T *is* $\langle\{a,b\}\{c\}\rangle$. *Thus, The similarity between two sequences becomes either* $sim_{LCS_{size}}(S,T) = \frac{2}{3}$ *or* $sim_{LCS_{length}}(S,T) = \frac{3}{6} = \frac{1}{2}$.

One may conclude that the use of the length or the size is decided by the data analyst based on his data set and prior knowledge. However, this duality between the size and length holds deeper consequences as the measure $sim_{LCS_{size}}$ does not satisfy the constraint concerning the identity of indiscernibles (i.e., two sequences may have a similarity of 1 while being different as shown in Figure 4.1.a and thus may hinder the following analysis processes).

Our measure avoids the use of an extra parameter of length or size of a sequence. The definition of sim_{ACS} depends only on the number of common subsequences and the total number of subsequences.

Another important point to highlight is the *sensitivity*. Our measure is very sensitive to sequence modifications when adding a new items or new itemsets. Let us, consider three sequences $U = \langle\{a,b,c\}\{a,b,c\}\{a,b,c\}\rangle$, $V = \langle\{a,b\}\{a,b\}\ \{a,b\}\rangle$ and $W = \langle\{a,b\}\{a,b\}\{a,b,\ d,e\}\rangle$. The similarity between U and V is equal to the similarity between U and W for all the three measures sim_{ACS}, $sim_{LCS_{length}}$ and $sim_{LCS_{size}}$. Thus, U is equidistant to V and W as shown in Figure 4.1.a.

When adding items d and e to the second itemset of U and recomputing the similarities, on can see that U remains equidistant to V and W, whether $sim_{LCS_{length}}$ or $sim_{LCS_{size}}$ is used. However, if sim_{ACS} is used, U appears to be more similar to W than to V, as illustrated in Figure 4.1.b. It is the same case when we add new itemset $\{d,e\}$ to the sequence U and recompute the similarities. We can see that U remains equidistant to V and W, whether $sim_{LCS_{length}}$ or $sim_{LCS_{size}}$ is used. When we use sim_{ACS}, U appears to be more similar to W than to V, as illustrated in Figure 4.2.c.

The rest of the chapter, up to the experiments section, will be dedicated to devise efficient techniques for computing $\phi(S)$ and $\phi(S,T)$.

4.5 Counting All Distinct Subsequences

In this section, we present an efficient technique *to count* the number, $\phi(S)$, of all distinct subsequences for a given sequence S. First, we present the intuition behind the proposed counting scheme. Suppose that we extend a given sequence $S = \langle X_1 \cdots X_n\rangle$ with an itemset Y and we observe the relation between $\phi(S)$ and $\phi(S \circ Y)$. Two cases may appear:

1. Y is disjoint with any itemset in S; i.e., for all $i = 1 \dots n$, $Y \cap S[i] = \varnothing$, then the number of distinct subsequences of $S \circ Y$ equals $|\varphi(S)| \cdot 2^{|Y|}$, since for all $T \in \phi(S)$ and $Y' \in \mathcal{P}_{\geqslant 1}(Y)$, $T \circ Y'$ is not in $\phi(S)$. For example, $\phi(\langle\{a,b\}\{c\}\rangle \circ \{d,e\}) = 8 \cdot 2^2 = 32$.

2. At least one item of Y appears in an itemset of S; i.e., $\exists i \in [1,n] : Y \cap S[i] \neq \varnothing$. In this case, $|\varphi(S \circ X)|$ is smaller than $|\varphi(S)| \cdot 2^{|Y|}$, because not every combination of a sequence in $\varphi(S)$ with an element from the power set of Y results in a unique subsequence. For example, if $S = \langle\{a,b\}\rangle$ and $Y = \{a,b\}$, the set of all subsequences of S is $\varphi(S) = \{\langle\rangle, \langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{a,b\}\rangle\}$ and the power set of Y is $\mathcal{P}(Y) = \{\varnothing, \{a\}, \{b\}, \{a,b\}\}$. The sequence $\langle\{a\}\rangle$ can be obtained by either extending the empty sequence $\langle\rangle \in \varphi(S)$ with the itemset $\{a\} \in \mathcal{P}(Y)$, or by extending $\langle\{a\}\rangle \in \varphi(S)$ with $\varnothing \in \mathcal{P}(Y)$.

   Therefore, we need to define a method to remove the repetitions from the count. Formally, $|\varphi(S \circ Y)| = |\varphi(S)| \cdot 2^{|Y|} - R(S,Y)$ where $R(S,Y)$ represents a *correction term* that equals the number of repetitions of subsequences that should be suppressed for a given S concatenated with the itemset Y.

We illustrate the second case with an example.

Example 35.*: Consider sequence S_4 from our toy data set. $S_4^2 = \langle\{a\}\{a,b,d\}\rangle$ is the 2-prefix of S_4. Recall from Example 33 that the total number of subsequences of S_4^2 is $\phi(S_4^2) = 15$. Now suppose that we extend this sequence S_4^2 with the itemset $Y = \{a,b,c\}$. Clearly, concatenating each sequence from $\varphi(S_4^2)$ with each element in the power set of $\{a,b,c\}$ will generate some subsequences multiple times. For instance, the subsequence $\langle\{a\}\{b\}\rangle$ is generated twice: $\langle\{a\}\rangle \circ \{b\}$ and $\langle\{a\}\{b\}\rangle \circ \varnothing$. The same applies to other subsequences $\langle\{a\}\rangle$, $\langle\{b\}\rangle$, $\langle\{a,b\}\rangle$, $\langle\{a\}\{a\}\rangle$ and $\langle\{a\}\{ab\}\rangle$. Thus, making a total of 6 subsequences that are counted twice. In this case, the correct number of distinct subsequences for $S_4^2 \circ Y = \langle\{a\}\{a,b,d\}\{a,b,c\}\rangle$ is $|\varphi(S_4^2)| \cdot 2^{|Y|} - R(S_4^2, Y) = 15 \cdot 2^3 - 6 = 114$.*

As illustrated by the above example, the difficulty lies in the computation of the value of the *correction term* $R(S,Y)$. The general idea is to compensate the repeated concatenation of subsequences from S by the power set of Y. The problem occurs with sequences in $\varphi(S) \circ \mathcal{P}_{\geqslant 1}(Y)$ that are already in $\varphi(S)$. Suppose T is such a sequence, then T must be decomposable as $T' \circ Y'$, where $T' \in \varphi(S^i)$ for some $i = 0 \dots n-1$, and $Y' \subseteq Y \cap S[j]$, for some $j \in i+1 \dots n$. The following definition introduces the *position set* that will index the itemset positions that generate duplicates in S.

Definition 62 (Position set)**.** *Given a sequence $S = \langle X_1 \cdots X_n\rangle$ and an itemset Y, $L(S,Y)$ is the set of all **maximal positions** where the itemset Y has a maximal intersection with the different itemsets $S[i]$, $i = 1 \dots n$. Formally,*

$$L(S,Y) = \{i \mid S[i] \cap Y \neq \varnothing \wedge \forall j; j > i \wedge S[i] \cap Y \nsubseteq S[j] \cap Y\}$$

Notice that if there are multiple positions that generate the same duplicates, we only consider the last one (right-most in the sequence).

Example 36.*: Consider $S_4^3 = \langle\{a\}\{a,b,d\}\{a,b,c\}\rangle$ from our running example. $L(\langle\{a\}\{a,b,d\}\{a,b,c\}\rangle, \{b,d\}) = \{2,3\}$. The index 2 is chosen because the intersection between $S_4[2]$ and $\{b,d\}$ is not empty (i.e., $\{a,b,d\} \cap \{b,d\} = \{b,d\} \neq \varnothing$) and is maximal in the sense that $S_4[2] \cap \{b,d\} \nsubseteq S_4[3] \cap \{b,d\}$. Similarly, the index 3 is chosen because $S_4[3] \cap \{b,d\} = \{b,d\} \neq \varnothing$ and it is the last itemset in the sequence.*

The following lemma now formalizes the observation that we only need to consider the sets $S[i]$ for i in the position set.

Lemma 1. *Let S be a sequence and Y an itemset. Then $\phi(S \circ Y) = \phi(S) \cdot 2^{|Y|} - R(S, Y)$, with*

$$R(S,Y) \;=\; \left| \bigcup_{\ell \in L} \left\{ \varphi(S^{\ell-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap Y) \right\} \right|$$

Proof. Let $T = \langle T_1, \ldots, T_m \rangle$ be a sequence that is counted multiple times; i.e., $T \in (\varphi(S) \circ \mathcal{P}_{\geqslant 1}(Y)) \cap \varphi(S)$. Clearly $T_m \in \mathcal{P}_{\geqslant 1}(Y)$ as otherwise T would not have been in $\varphi(S) \circ \mathcal{P}_{\geqslant 1}(Y)$. Let ℓ denote $\max\{j | T_m \subseteq S[j]\}$. Since $T \in \varphi(S)$, such ℓ must exist. Then, $\ell \in L(S, Y)$, since ℓ is the largest index for which $S[\ell] \cap Y$ includes T_m. Therefore, $T \in S^{\ell-1} \circ (\mathcal{P}_{\geqslant 1}(S[\ell] \cap Y))$ for a $\ell \in L(S, Y)$. □

Notice, however, that the sets $\varphi(S^{\ell-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap Y)$ are not necessarily disjoint; consider, e.g., $S = \langle \{a, b\}\{b, c\} \rangle$ and $Y = \{a, b, c\}$. The position set is $L = \{1, 2\}$, and $\langle \{b\} \rangle$ appear in both $\varphi(S^0) \circ \mathcal{P}_{\geqslant 1}(S[1] \cap Y)$ and $\varphi(S^1) \circ \mathcal{P}_{\geqslant 1}(S[2] \cap Y)$. To incorporate this overlap, we compute the cardinality of the union in Lemma 1 using the inclusion-exclusion principle, leading to the following theorem:

Theorem 3. *Let $S = \langle X_1 \ldots X_n \rangle$ and Y be an itemset. Then,*

$$\phi(S \circ Y) = \phi(S) \cdot 2^{|Y|} - R(S, Y) \tag{4.1}$$

with

$$R(S,Y) = \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left(\phi(S^{\min(K)-1}) \cdot \left(2^{|\left(\bigcap_{j \in K} S[j]\right) \cap Y|} - 1 \right) \right) \tag{4.2}$$

Proof. The proof is a simple application of the inclusion-exclusion principle to compute the cardinality of the union of Lemma 1:

$$R(S,Y) = \left| \bigcup_{\ell \in L(S,Y)} \left\{ \varphi(S^{\ell-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap Y) \right\} \right|$$

$$R(S,Y) = \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left| \bigcap_{\ell \in K} \left\{ \varphi(S^{\ell-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap Y) \right\} \right|$$

The proof is completed by the following two observations:

$$\bigcap_{\ell \in K} \left\{ \varphi(S^{\ell-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap Y) \right\} \;=\; \varphi(S^{\min(K)-1}) \circ \mathcal{P}_{\geqslant 1}((\cap_{k \in K} S[k]) \cap Y)$$

Indeed; any sequence of length m in set_K has $T^{m-1} \in S^{\min(K)-1}$, and $T_m \in \mathcal{P}_{\geqslant 1}(S[k] \cap Y)$, for all $k \in K$. And, the second observation:

$$\left| \varphi(S^{\min(K)-1}) \circ \mathcal{P}_{\geqslant 1}((\cap_{k \in K} S[k]) \cap Y) \right| = \phi(S^{\min(K)-1}) \cdot \left(2^{|(\cap_{k \in K} S[k]) \cap Y|} - 1 \right)$$

□

Example 37.: *We illustrate the complete counting process with sequence S_4^3. The position sets for this sequence are:* $L(\langle\rangle, \{a\}) = \varnothing,\ L(\langle\{a\}\rangle, \{a,b,d\}) = \{1\},\ L(\langle\{a\}\{a,b,d\}\rangle, \{a,b,c\}) = \{2\}$, $L(\langle\{a\}\{a,b,d\}\{a,b,c\}\rangle, \{b,d\}) = \{2,3\}$.

$$\begin{aligned}
\phi(\langle\rangle) &= 1 \\
\phi(\langle\{a\}\rangle) &= \phi(\langle\rangle)\cdot 2^{|\{a\}|} = 2 \\
\phi(\langle\{a\}\{a,b,d\}\rangle) &= \phi(\langle\{a\}\rangle)\cdot 2^{|\{a,b,d\}|} - \phi(\langle\rangle)\cdot(2^{|\{a,b,d\}\cap\{a\}|}-1) \\
&= 2\cdot 2^3 - 1\cdot(2^1-1) = 15 \\
\phi(\langle\{a\}\{a,b,d\}\{a,b,c\}\rangle) &= \phi(\langle\{a\}\{a,b,d\}\rangle)\cdot 2^{|\{a,b,c\}|} - \phi(\langle\{a\}\rangle)\cdot(2^{|\{a,b,d\}\cap\{a,b,c\}|}-1) \\
&= 15\cdot 2^3 - 2\cdot(2^2-1) = 114
\end{aligned}$$

4.6 Counting All Common Subsequences

In this section, we will extend the previous results to count all common distinct subsequences between two sequences S and T. Again, we discuss the basic intuition and then present the main result. Suppose that we extend the sequence S with an itemset Y and we observe the relation between $\varphi(S,T)$ and $\varphi(S\circ Y,T)$, two cases may appear:

1. If no items in Y appear in any itemset of S and T then the concatenation of the itemset Y with the sequence S *has no effect* on the the set $\varphi(S,T)$.

2. If at least one item in Y appears in either one of the sequences S or T (or both) then it can be observed that new common subsequences may appear in $\varphi(S,T)$. As for the counting method of the distinct subsequences of a unique sequence S, repetitions may occur and a generalized correction term for both S and T needs to be defined. Formally,

$$|\varphi(S\circ Y,T)| = |\varphi(S,T)| + A(S,T,Y) - R(S,T,Y)$$

   where $A(S,T,Y)$ represents the number of extra common subsequences that should be added and $R(S,T,Y)$ is the correction term.

Similarly to the distinct subsequence problem, the position set will index the positions that generate duplicate sequences. The following lemma formalizes this observation:

Lemma 2. *Let $S=\langle X_1\dots X_n\rangle$, $T=\langle X'_1\dots X'_m\rangle$ and Y an itemset.*

$$A(S,T,Y) = \left|\bigcup_{\ell\in L(T,Y)}\left\{\varphi(S,T^{\ell-1})\circ\mathcal{P}_{\geqslant 1}(T[\ell]\cap Y)\right\}\right|$$

$$R(S,T,Y) = \left|\bigcup_{\ell\in L(S,Y)}\left\{\bigcup_{\ell'\in L(T,Y)}\left\{\varphi(S^{\ell-1},T^{\ell'-1})\circ\mathcal{P}_{\geqslant 1}(S[\ell]\cap T[\ell']\cap Y)\right\}\right\}\right|$$

Proof. Let $Z=\langle Z_1,\dots,Z_m\rangle$ be a new subsequence that is added to $\varphi(S,T)$ after concatenating sequence S with itemset Y; i.e., $Z\in\varphi(S\circ Y,T)\backslash\varphi(S,T)$. Clearly $Z_m\in\mathcal{P}_{\geqslant 1}(Y)$ as otherwise Z would not have been added to $\varphi(S\circ Y,T)$. Let $\ell' = \max\{j|Z_m\subseteq T[j]\}$. Since $Z\in\varphi(S\circ Y,T)$,

with $Z \preceq T$, then such $\ell' \in L(T,Y)$ must exist. ℓ' is the largest index for which $T[\ell'] \cap Y$ includes Z_m. Therefore, $Z \in \varphi(S, T^{\ell'-1}) \circ (\mathcal{P}_{\geqslant 1}(T[\ell'] \cap Y))$ for a $\ell' \in L(T,Y)$.

Let $W = \langle W_1, \ldots, W_m \rangle$ be a sequence that is counted multiple times; i.e., $W \in (\varphi(S, T^{\ell'-1}) \circ \mathcal{P}_{\geqslant 1}(T[\ell'] \cap Y)) \cap \varphi(S,T)$ where $\ell' \in L(T,Y)$. Clearly $W_m \in \mathcal{P}_{\geqslant 1}(T[\ell'] \cap Y)$ as otherwise W would not have been in $\varphi(S, T^{\ell'-1}) \circ \mathcal{P}_{\geqslant 1}(T[\ell'] \cap Y)$. Let $\ell = \max\{j | W_m \subseteq S[j]\}$. Since $W \in \varphi(S,T)$, such $\ell \in L(S,Y)$ must exist, since ℓ is the largest index for which $S[\ell] \cap Y$ includes Z_m. Therefore, $Z \in \varphi(S^{\ell-1}, T^{\ell'-1}) \circ (\mathcal{P}_{\geqslant 1}(S[\ell] \cap T[\ell'] \cap Y))$ for $\ell \in L(S,Y)$ and $\ell' \in L(T,Y)$. □

Example 38.: *Let $S_1^4 = \langle \{a\}\{a,b\}\{e\}\{c,d\} \rangle$ be the 4-prefix of S_1, and let $S_2^3 = \langle \{a\}\{b,c,d\}\{a,d\} \rangle$ be the 3-prefix of S_2 from our running example. Suppose that we extend S_1^4 with the itemset $Y = \{b,d\}$ and count all distinct common subsequences between $S_1^4 \circ \{b,d\}$ and S_2^3. Notice that the intersections between itemset $\{b,d\}$ and the itemsets in S_2^3 are non-empty and maximal for the itemsets $\{b,c,d\}$ and $\{a,d\}$. Thus, $L(S_2^3, \{b,d\}) = \{2,3\}$ and $A(S_1^4, S_2^3, \{b,d\}) = |\{\varphi(S_1^4, S_2^1) \circ \mathcal{P}_{\geqslant 1}(\{b,d\} \cap \{b,c,d\})\} \cup \{\varphi(S_1^4, S_2^2) \circ \mathcal{P}_{\geqslant 1}(\{b,d\} \cap \{a,d\})\}| = 14$.*

In a similar way, the intersections between itemset $\{b,d\}$ and the itemsets in S_1^4 are non-empty and maximal for the itemsets in positions 2 and 4. Thus $L(S_1^4, \{b,d\}) = \{2,4\}$. In this case, adding the values $A(S_1^4, S_2^3, \{b,d\})$ to $\phi(S_1^4, S_2^3)$ will count erroneously some subsequences.

For instance, the subsequences $\langle \{a\}\{b\}\{d\} \rangle$ and $\langle \{b\}\{d\} \rangle$ are counted twice: once in $\varphi(S_1^4, S_2^3)$ and the other when all sequences of the set $\varphi(S_1^4, S_2^3)$ are extended with $\{b,d\} \cap \{a,d\}$. The same remark applies to other subsequences: $\langle \{b\} \rangle$,$\langle \{d\} \rangle$, $\langle \{a\}\{b\} \rangle$ and $\langle \{a\}\{d\} \rangle$. In this case, the correct number of all common distinct subsequences between $S_1^4 \circ \{b,d\}$ and S_2^3 is $|\varphi(S_1^4, S_2^2)| + A(S_1^4, S_2^3, \{b,d\}) - R(S_1^4, S_2^3, \{b,d\})$ where:

$$\begin{aligned} R(S_1^4, S_2^3, \{b,d\}) &= | \quad \{\varphi(S_1^1, S_2^1) \circ \mathcal{P}_{\geqslant 1}(\{a,b\} \cap \{b,c,d\} \cap \{b,d\})\} \\ &\quad \cup \{\varphi(S_1^1, S_2^2) \circ \mathcal{P}_{\geqslant 1}(\{a,b\} \cap \{a,d\} \cap \{b,d\})\} \\ &\quad \cup \{\varphi(S_1^3, S_2^1) \circ \mathcal{P}_{\geqslant 1}(\{c,d\} \cap \{b,c,d\} \cap \{b,d\})\} \\ &\quad \cup \{\varphi(S_1^3, S_2^2) \circ \mathcal{P}_{\geqslant 1}(\{c,d\} \cap \{a,d\} \cap \{b,d\})\}| \\ &= \quad 6 \end{aligned}$$

Thus,

$$\begin{aligned} \phi(S_1^4 \circ \{b,d\}, S_2^3) &= |\varphi(S_1^4, S_2^2)| + A(S_1^4, S_2^3, \{b,d\}) - R(S_1^4, S_2^3, \{b,d\}) \\ &= 13 + 14 - 6 = 21 \end{aligned}$$

Similarly to Lemma 1 and as illustrated in the above example, the computation of the cardinality of the unions in Lemma 2 implies the usage of the inclusion-exclusion principle. This remark leads to the second theorem:

Theorem 4. *Let $S = \langle X_1 \ldots X_n \rangle$, $T = \langle X'_1 \ldots X'_m \rangle$ and Y an itemset. Then,*

$$\phi(S \circ Y, T) = \phi(S,T) + A(S,T,Y) - R(S,T,Y) \tag{4.3}$$

with

$$A(S,T,Y) = \sum_{K \subseteq L(T,Y)} (-1)^{|K|+1} \left(\phi(S, T^{\min(K)-1}) \cdot \left(2^{|(\bigcap_{j \in K} T[j]) \cap Y|} - 1 \right) \right) \tag{4.4}$$

and

$$R(S,T,Y) = \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left(\sum_{K' \subseteq L(T,Y)} (-1)^{|K'|+1} \cdot f(K,K') \right) \tag{4.5}$$

where

$$f(K,K') = \phi(S^{\min(K)-1}, T^{\min(K')-1}) \cdot \left(2^{|(\bigcap_{j \in K} S[j]) \cap \left(\bigcap_{j' \in K'} T[j']\right) \cap Y|} - 1 \right)$$

Proof. **Case 1:** No items in Y appear in any itemset of S and T, in this case the set of all common distinct subsequences between $S \circ Y$ and T is exactly the same set of all common distinct subsequences between S and T. Hence, $\phi(S \circ Y, T) = \phi(S,T)$.

Case 2: If at least an item in Y appears in either one of the sequences S or T (or both), then $\varphi(S \circ Y, T)$ is expressed as the union of the set of all common distinct subsequences between S and T with the set of added sequences $\mathcal{A}$ *without* the set of repeated sequences $\mathcal{R}$. Formally,

$$\varphi(S \circ Y, T) = \varphi(S,T) \cup \mathcal{A} \backslash \mathcal{R} \tag{4.6}$$

with

$$\mathcal{A} = \left\{ \bigcup_{\ell' \in L(T,Y)} \varphi(S, T^{\ell'-1}) \circ \mathcal{P}_{\geqslant 1}(T[\ell'] \cap Y) \right\}$$

$$\mathcal{R} = \left\{ \bigcup_{\ell \in L(S,Y)} \left\{ \bigcup_{\ell' \in L(T,Y)} \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap T[\ell'] \cap Y) \right\} \right\}$$

Notice that because these three sets are disjoint, the cardinality of $\varphi(S \circ Y, T)$ can be simply expressed as $|\varphi(S \circ Y, T)| = |\varphi(S,T)| + |\mathcal{A}| - |\mathcal{R}|$. Using the inclusion-exclusion principle, $|\mathcal{A}|$, denoted as $A(S,T,Y)$ can be written as,

$$\begin{aligned} A(S,T,Y) &= \left| \left\{ \bigcup_{\ell \in L(T,Y)} \varphi(S, T^{\ell-1}) \circ \mathcal{P}_{\geqslant 1}(T[\ell] \cap Y) \right\} \right| \\ &= \sum_{K \subseteq L(T,Y)} (-1)^{|K|+1} \left| \bigcap_{\ell \in K} \left\{ \varphi(S, T^{\ell-1}) \circ \mathcal{P}_{\geqslant 1}(T[\ell] \cap Y) \right\} \right| \end{aligned} \tag{4.7}$$

$A(S,T,Y)$ is completed by the following two observations:

$$\begin{aligned} set_K &:= \bigcap_{\ell \in K} \left\{ \varphi(S, T^{\ell-1}) \circ \mathcal{P}_{\geqslant 1}(T[\ell] \cap Y) \right\} \\ &= \varphi(S, T^{\min(K)-1}) \circ \mathcal{P}_{\geqslant 1}((\cap_{k \in K} T[k]) \cap Y) \end{aligned}$$

And, the second observation:

$$|set_K| = \phi(S, T^{\min(K)-1}) \cdot \left(2^{|(\cap_{k \in K} T[k]) \cap Y|} - 1 \right)$$

$A(S,T,Y)$ can be written as,

$$A(S,T,Y) = \sum_{K \subseteq L(T,Y)} (-1)^{|K|+1} \left(\phi(S, T^{\min(K)-1}) \cdot \left(2^{|\left(\bigcap_{j \in K} T[j]\right) \cap Y|} - 1 \right) \right)$$

The same inclusion-exclusion reasoning applies to the cardinality of $\mathcal{R}$, denoted $R(S,T,Y)$

$$R(S,T,Y) = \left| \left\{ \bigcup_{\ell \in L(S,Y)} \left\{ \bigcup_{\ell' \in L(T,Y)} \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap T[\ell'] \cap Y) \right\} \right\} \right|$$

$$= \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left(\sum_{K' \subseteq L(T,Y)} (-1)^{|K'|+1} \left| \bigcap_{\ell \in K} \bigcap_{\ell' \in K'} \left\{ \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap T[\ell'] \cap Y) \right\} \right| \right)$$

The final result follows after noticing that,

$$\begin{aligned} set_{K,K'} &= \bigcap_{\ell \in K} \bigcap_{\ell' \in K'} \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geqslant 1}(S[\ell] \cap T[\ell'] \cap Y) \\ set_{K,K'} &= \varphi(S^{\min(K)-1}, T^{\min(K')-1}) \circ \mathcal{P}_{\geqslant 1}((\cap_{k \in K} S[k]) \cap \left(\cap_{k' \in K'} T[k'] \right) \cap Y) \end{aligned}$$

$R(S,T,Y)$ can be written as,

$$R(S,T,Y) = \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left(\sum_{K' \subseteq L(T,Y)} (-1)^{|K'|+1} \cdot f(K,K') \right)$$

where:

$$f(K,K') = \phi(S^{\min(K)-1}, T^{\min(K')-1}) \cdot \left(2^{|\left(\bigcap_{j \in K} S[j]\right) \cap \left(\bigcap_{j' \in K'} T[j']\right) \cap Y|} - 1 \right)$$

□

4.6.1 Dynamic Programming

Theorem 4 leads to a simple dynamic programming algorithm. For two given sequences S and T, such that $|S| = n$ and $|T| = m$, the program produces a $(n+1) \times (m+1)$ matrix (with indices starting at 0), denoted $\mathcal{M}$ such that:

$$\mathcal{M}_{i,j} = \begin{cases} 1 & i = 0 \\ 1 & j = 0 \\ \phi(S^i, T^j) & \text{otherwise.} \end{cases}$$

Example 39.: *Consider the two sequences S_1 and S_2 from our running example. $\phi(S_1, S_2) = 21$ and the set of all common subsequences of S_1 and S_2 is:*

$$\begin{aligned} \varphi(S_1, S_2) = \{ &\langle\rangle, \langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{c\}\rangle, \langle\{d\}\rangle, \langle\{c,d\}\rangle, \langle\{b,d\}\rangle, \langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle, \\ &\langle\{a\}\{d\}\rangle, \langle\{a\}\{b,d\}\rangle, \langle\{a\}\{c,d\}\rangle, \langle\{b\}\{d\}\rangle, \langle\{c\}\{d\}\rangle, \langle\{d\}\{d\}\rangle, \langle\{c,d\}\{d\}\rangle, \\ &\langle\{a\}\{d\}\{d\}\rangle, \langle\{a\}\{b\}\{d\}\rangle, \langle\{a\}\{c\}\{d\}\rangle, \langle\{a\}\{cd\}\{d\}\rangle\} \end{aligned}$$

We detail the computation of the cell $\mathcal{M}_{2,3}$ which represents the number of all common subsequence between $S_1^1 \circ \{a,b\}$ and S_2^3. The position sets are $L(S_1^1, \{a,b\}) = \{1\}$ and $L(S_2^3, \{a,b\}) = \{2,3\}$. Using the result in Theorem 4:

$$\begin{aligned}\mathcal{M}_{2,3} &= \phi(\langle\{a\}\{a,b\}\rangle, \langle\{a\}\{b,c,d\}\{a,d\}\rangle) \\ &= \phi(\langle\{a\}\rangle, \langle\{a\}\{b,c,d\}\{a,d\}\rangle) + A(S_1^1, S_2^3, \{a,b\}) - R(S_1^1, S_2^3, \{a,b\})\end{aligned}$$

$\phi(\langle\{a\}\rangle, \langle\{a\}\{b,c,d\}\{a,d\}\rangle)$ was previously computed and stored in $\mathcal{M}_{1,3} = 2$. $A(S_1^1, S_2^3, \{a,b\})$ and $R(S_1^1, S_2^3, \{a,b\})$ are computed as follows:

$$\begin{aligned}A(S_1^1, S_2^3, \{a,b\}) &= \phi(S_1^1, S_2^1) \cdot (2^{|\{b,c,d\} \cap \{a,b\}|} - 1) + \phi(S_1^1, S_2^2) \cdot (2^{|\{a,d\} \cap \{a,b\}|} - 1) \\ &\quad -\phi(S_1^1, S_2^1) \cdot (2^{|\{b,c,d\} \cap \{a,d\} \cap \{a,b\}|} - 1) \\ &= \mathcal{M}_{1,1} \cdot (2-1) + \mathcal{M}_{1,2} \cdot (2-1) - \mathcal{M}_{1,1} \cdot (1-1) \\ &= 2 + 2 - 0 = 4\end{aligned}$$

$$\begin{aligned}R(S_1^1, S_2^3, \{a,b\}) &= \phi(S_1^0, S_2^1) \cdot (2^{|\{a\} \cap \{b,c,d\} \cap \{a,b\}|} - 1) \\ &\quad +\phi(S_1^0, S_2^2) \cdot (2^{|\{a\} \cap \{a,d\} \cap \{a,b\}|} - 1) \\ &\quad -\phi(S_1^0, S_2^1) \cdot (2^{|\{a\} \cap \{b,c,d\} \cap \{a,d\} \cap \{a,b\}|} - 1) \\ &= \mathcal{M}_{0,1} \cdot (1-1) + \mathcal{M}_{0,2} \cdot (2-1) - \mathcal{M}_{0,1} \cdot (1-1) = 1\end{aligned}$$

Finally, $\mathcal{M}_{2,3} = \phi(\langle\{a\}\{a,b\}\rangle, \langle\{a\}\{b,c,d\}\{a,d\}\rangle) = 2 + 4 - 1 = 5$.
The entire computation for $\phi(S_1, S_2)$ is illustrated in Table 4.2.

	$\{\varnothing\}$	$\{a\}$	$\{b,c,d\}$	$\{a,d\}$
$\{\varnothing\}$	1	1	1	1
$\{a\}$	1	2	2	2
$\{a,b\}$	1	2	4	5
$\{e\}$	1	2	4	5
$\{c,d\}$	1	2	10	13
$\{b,d\}$	1	2	12	21

Table 4.2: Matrix for counting all common subsequences between S_1 and S_2

4.7 Complexity and Linial-Nisan Approximation Results

4.7.1 Complexity

We will now discuss the complexity of computing the number of subsequences in a sequence of itemets and the number of common subsequences in two such sequences using the formulas in Theorems 3 and 4. Essential in this analysis is the size of the position set $L(S,Y)$, which will highly depend on the specific cases. It is important to notice that the size of $L(S,Y)$ is bounded

by both $2^{|Y|}$ (every index corresponds to a unique subset of Y) and $|S|$ (every index corresponds to a unique position within S). Notice incidentally that the worst case $|L(S^{\ell-1}, S[\ell])| = \ell - 1$ is unlikely to happen for long sequences, as this implies that if we construct the following sequence: $S[1] \cap S[\ell], \ldots, S[\ell-1] \cap S[\ell]$, none of the entries in the sequence is followed by a superset. This would only happen in pathological cases such as $\langle\{a,b,c\}\{a,b\}\{a,c\}\{b,c\}\{a\}\{b\}\{c\}\{a,b,c\}\rangle$. In this case $L(S^{\ell-1}, S[\ell]) = \{1,2,3,4,5,6,7,8\}$.

First we analyze the complexity of the brute-force method consisting of generating all subsequences followed by elimination of duplicates. For a sequence $\langle S_1 ... S_\ell \rangle$, there are at most $N = \Pi_{i=1}^{\ell} 2^{|S[i]|} = 2^{\ell(S)}$ subsequences we need to consider; for every position $i = 1 \ldots \ell$, any subset of S_i needs to be considered in combination with every subset of the other positions. Eliminating the duplicates can be done in time $N \log(N)$. The total complexity is hence $\mathcal{O}(N \log(N))$. Hence the total complexity of the brute-force method for finding all subsequences of S is bounded by $\ell(S)2^{\ell(S)}$. When computing the number of subsequences that two sequences S and T have in common, first their respective subsequences are listed, sorted and deduplicated. Then the two ordered lists can be intersected in time linear in the length of the longest list. Let M be the upper bound on the number of subsequences of T. The complexity of these operations comes down to $\mathcal{O}(N \log(N) + M \log(M)) = \mathcal{O}(\ell(S)2^{\ell(S)} + \ell(T)2^{\ell(T)})$.

Let us now analyze the complexity of the computation based upon the formula in Theorem 4. Suppose that we need to compute the number of subsequences of $\langle S_1 ... S_\ell \rangle$. Assume that we know the number of unique subsequences for all $S^k, k < \ell$. The number of computations we need to perform if we apply the formula of Theorem 3, to get the number of subsequences of $\langle S_1 ... S_k \rangle$ is proportional to the size of the powerset of $L(S^{\ell-1}, S[\ell])$; indeed, we need to compute a sum over all subsets of $L(S^{\ell-1}, S[\ell])$. It is easy to see that the size of this set $L(S^{\ell-1}, S[\ell])$ is bounded by $\min(\ell - 1, 2^{|S[\ell]|})$: every position can appear at most once, and for every subset in $S[\ell]$ there can be at most one position. The total complexity is therefore bounded by $\sum_{k=1}^{\ell} 2^{\min(k-1, 2^{|S[k]|})}$ and $\sum_{k=1}^{\ell} 2^{k-1} = 2^\ell - 1 = \mathcal{O}(2^\ell)$ is an upper bound, which is significantly better than the brute-force method listing all subsequences and removing the duplicates; even though $\mathcal{O}(2^\ell)$ is a very conservative upper bound (position sets will usually only contain a fraction of all possible indices!), the difference between 2^ℓ and $2^{\ell(S)}$ is enormous if the individual itemsets are large. For the pathological case described in the beginning of this section, for instance, the difference is as big as $2^{\ell(S)} = 2^{32}$ versus $2^\ell = 2^{15}$; that is: the number of steps by the method based on Theorem 3 is 131 072 times smaller than in the brute force algorithm.

The complexity for the computation of the number of common subsequences in Theorem 4 goes along the same lines. Again we will first assume that for two sequences S and T, the number of common subsequences $\phi(S^i, T^j)$ has been computed for all S^i and T^j with (i,j) smaller than $(|S|, |T|)$. Let Y be the last itemset of S; that is, $Y = S[|S|]$, and $S' = S^{|S|-1}$. The main complexity term in the formula in Theorem 4 is in $R(S', T, S[|S|])$; this term dominates the complete expression in terms of computational complexity. The complexity of the double sum is proportional to $2^{|L(S',Y)|}2^{|L(T,Y)|}$, which is bounded by $2^{\min(|S'|, 2^{|Y|})}2^{\min(|T|, 2^{|Y|})} < 2^{|S'||T|}$. So, the total complexity, taking into account that we need to compute the number of common subsequences for all subsequences of S and T as well (cfr. the dynamic programming approach given in 4.6.1), leads to a total complexity of $\sum_{i=1}^{|S|} \sum_{j=1}^{|T|} 2^{ij} = \mathcal{O}(\min(|S|, |T|) \cdot 2^{|S||T|})$.

In conclusion, both the brute-force and the dynamic approach are, in worst case, exponential in the input. The brute-force method, however, is exponential in $\ell(S)$ and $\ell(T)$, whereas the dynamic approach is only exponential in the position sets L, which *can be* (but only in pathological cases *are*) as large as $|S| << |\ell(S)|$ and $|T| << \ell(T)$. So, even in the very unlikely worst case the dynamic approach is much better than the brute-force method. The brute-force algorithm,

in contrast, takes exponential time in all cases.

4.7.2 Linial-Nisan Approximation

As stated by Linial and Nisan [1990]: *"Many computational problems may be viewed as asking for the size of a union of a collection of sets. [...] In these cases, the inclusion-exclusion formula may be used to find the size of the union"*. Our similarity measure relies heavily on the inclusion-exclusion principle: on the one hand, the exact computation of the number of all distinct subsequences of a sequence requires the computation of the *correction number*, $R(S,Y)$ in Equation 4.2, on the other hand the number of common subsequences needs the computation of the addition and correction terms, $A(S,T,Y)$ and $R(S,T,Y)$ in Equations 4.4 and 4.5. The computation drawback is the fact that the inclusion-exclusion formula has an exponential number of terms which, as mentioned previously in the complexity subsection, can become a problem with very long sequences and a position set L of large cardinality. This prompted our interest in approximating our similarity measure through the approximation of the inclusion-exclusion formula used in both ACS and ADS computations.

Theorem 5 (Linial-Nisan Approximation)**.** *[Linial and Nisan, 1990, Theorem 2]. Let $A_1, A_2, \ldots, A_N$ be a collection of sets. Suppose that $|\bigcap_{i\in S} A_i|$ is given for every subset $S \subset [N]$ of cardinality $|S| < K$. For any integers K, N there exist (explicitly given) constants $(\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N})$ such that for every collection of sets $A_1, A_2, \ldots, A_N$, the quantity*

$$\sum_{|S|\leqslant K} \alpha_{|S|}^{K,N} |\bigcap_{i\in S} A_i|$$

differs from $|\bigcup_{i=1}^{N} A_i|$ by at most a factor of $1 + O(e^{-\frac{2K}{\sqrt{N}}})$ if $K \geqslant \Omega(\sqrt{N})$ or $O(\frac{N}{K^2})$ if $K \leqslant O(\sqrt{N})$.

The real numbers $\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N}$ are defined by Linial and Nisan to be the coefficients of the linearly transformed Chebyshev polynomials expressed in terms of the polynomials $\binom{x}{1}, \binom{x}{2}, \ldots, \binom{x}{K}$. $\overrightarrow{\alpha} = (\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N})$ is calculated efficiently by solving a set of linear equations. Consider the above polynomial identity for $x = 1, \ldots, K$. The vector of coefficients is calculated as follows: $\overrightarrow{\alpha} = \overrightarrow{t} \cdot \mathcal{A}^{-1}$, where $\mathcal{A}$ is the matrix whose (i,j) entry is $\binom{j}{i}$. The inverse matrix $\mathcal{A}^{-1}(i,j)$ is defined as $(-1)^{i+j}\binom{j}{i}$, $\overrightarrow{t} = (q_{K,N}(1), q_{K,N}(2), \ldots, q_{K,N}(K))$ is the linearly transformed Chebyshev polynomials, $q_{K,N}(x) = 1 - \frac{T_k(\frac{2x-(N+1)}{N-1})}{T_k(\frac{-(N+1)}{N-1})}$ and $T_K(x)$ is a polynomial of degree K and is given by

$$T_K(x) = \frac{(x + \sqrt{x^2-1})^K + (x - \sqrt{x^2-1})^K}{2}$$

In our approximation method, every time that the position set is too big (i.e., $|L| \geqslant \sigma$ where σ is a user provided size threshold) we compute $\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_k^{K,N}$ with $K = \lceil\sqrt{|L|}\rceil$ and $N = |L|$ and then approximate the inclusion-exclusion formula through the following Theorems:

Theorem 6. *Let $S = \langle X_1 \ldots X_n\rangle$ and Y an itemset. Then,*

$$\phi_{LN}(S \circ Y) = 2^{|Y|} \cdot \phi_{LN}(S) - R_{LN}(S,Y) \tag{4.8}$$

with

$$R_{LN}(S,Y) = \sum_{k=1}^{K} \alpha_k^{K,N} \sum_{\substack{O \subseteq L(S,Y) \\ |O|=k}} \phi_{LN}(S^{\min(O)-1}) \cdot \left(2^{|\left(\bigcap_{j\in O} S[j]\right)\cap Y|} - 1\right)$$

where $N = |L(S,Y)|$, $K = \lceil\sqrt{|N|}\rceil$ *and* $\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N}$ *are the Linial-Nisan coefficients.*

Theorem 7. *Let* $S = \langle X_1 \ldots X_n\rangle$, $T = \langle X'_1 \ldots X'_m\rangle$, Y *an itemset. Then,*

$$\phi_{LN}(S \circ Y, T) = \phi_{LN}(S,T) + A_{LN}(S,T,Y) - R_{LN}(S,T,Y) \tag{4.9}$$

with

$$A_{LN}(S,T,Y) = \sum_{k'=1}^{K'} \alpha_k^{'K',N'} \sum_{\substack{O' \subseteq L(T,Y) \\ |O'|=k'}} \phi_{LN}(S, T^{\min(O')-1})(2^{|\left(\bigcap_{j\in O'} T[j]\right)\cap Y|} - 1)$$

$$R(S,T,Y) = \sum_{k=1}^{K} \alpha_k^{K,N} \sum_{\substack{O \subseteq L(S,Y) \\ |O|=k}} \left(\sum_{k'=1}^{K'} \alpha_k^{'K',N'} \sum_{\substack{O' \subseteq L(S,Y) \\ |O'|=k'}} f(O,O') \right)$$

where
$f(O,O') = \phi_{LN}(S^{\min(O)-1}, T^{\min(O')-1}) \cdot \left(2^{|\left(\bigcap_{j\in O} S[j]\right)\cap\left(\bigcap_{j'\in O'} T[j']\right)\cap Y|} - 1\right)$, $N = |L(S,Y)|$, $N' = |L(T,Y)|$, $K = \lceil\sqrt{|L(S,Y)|}\rceil$, $K' = \lceil\sqrt{|L(T,Y)|}\rceil$ *and* $\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N}$, $\alpha_1^{'K',N'}, \alpha_2^{'K',N'}$ $, \ldots, \alpha_{K'}^{'K',N'}$ *are the Linial-Nisan coefficients.*

Example 40.*: Consider* $S = \langle\{a,c,d,e,f,g,h,i,j,k\}\{a,b,d,e,f,g,h,i,j,k\}\{a,b,c,e,f,g,h,i,$ $j,k\}\{a,b,c,d,f,g,h,i,j,k\}\{a,b,c,d,e,g,h,i,j,k\}\{a,b,c,d,e,f,h,i,j,k\}\{a,b,c,d,e,f,g,i,j,k\}\{a,$ $b,c,d,e,f,g,h,j,k\}\{a,b,c,d,e,f,g,h,i,k\}\{a,b,c,d,e,f,g,h,i,j,k\}\rangle$*. The number of distinct subsequences for* S^9 *is* $\phi(S^9) = 1{,}233\,117\,9 \cdot 10^{27}$ *and the position set for the last itemset is* $L(S^9,$ $S[10]) = \{1,2,3,4,5,6,7,8,9\}$*. With the normal exact computation the final number of distinct subsequences is:* $\phi(S^{10}) = 2^{|\{a,b,c,d,e,f,g,h,i,j,,k\}|} \cdot \phi(S^9) - R(S^9, S[10]) = 2{,}524\,192\,3 \cdot 10^{30}$*. Notice that the inclusion-exclusion formula used for the computation of* $R(S^9, S[10])$ *contains* $\sum_{i=1}^{9}\binom{9}{i}$ *terms.*

To do the Linial-Nisan approximation, remark that $N = |L| = 9$ *and* $K = \lceil\sqrt{N}\rceil = 3$*. The vector of Linial-Nisan coefficients is defined as* $\overrightarrow{\alpha} = (\alpha_1^{3,9}, \alpha_2^{3,9}, \alpha_3^{3,9}) = \overrightarrow{t} \cdot \mathcal{M}^{-1}$ *where* $\mathcal{A}$ *is the matrix whose* (i,j) *entry is* $\binom{j}{i}$*. The inverse matrix* $\mathcal{A}^{-1}(i,j)$ *is defined as* $(-1)^{i+j}\binom{j}{i}$*. In our example,*

$$\mathcal{M}^{-1} = \begin{pmatrix} 1 & -2 & 3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix}$$

$\overrightarrow{t} = (q_{K,N}(1), q_{K,N}(2), \ldots, q_{K,N}(K))$ *is the vector of linearly transformed Chebyshev polynomials and is computed using the polynomial* $T_K(x)$ *as follows:*

$$\begin{aligned}
q_{3,9}(1) &= 1 - \frac{T_3(\frac{2-(9+1)}{9-1})}{T_3(\frac{-(9+1)}{9-1})} = 1 - \frac{T_3(-1)}{T_3(-\frac{10}{8})} = 1 - \frac{-1}{-4,06} = 0.75 \\
q_{3,9}(2) &= 1 - \frac{T_3(\frac{4-(9+1)}{9-1})}{T_3(\frac{-(9+1)}{9-1})} = 1 - \frac{T_3(-\frac{6}{8})}{T_3(-\frac{10}{8})} = 1 - \frac{0.56}{-4,06} = 1.13 \\
q_{3,9}(3) &= 1 - \frac{T_3(\frac{6-(9+1)}{9-1})}{T_3(\frac{-(9+1)}{9-1})} = 1 - \frac{T_3(-\frac{4}{10})}{T_3(-\frac{10}{8})} = 1 - \frac{1}{-4,06} = 1.24
\end{aligned}$$

This vector is necessary to solve the system of linear equations:

$$\begin{aligned}
\overrightarrow{\alpha} &= (\alpha_1^{3,9}, \alpha_2^{3,9}, \alpha_3^{3,9}) = \overrightarrow{t} \cdot \mathcal{M}^{-1} \\
&= \begin{pmatrix} 0.75 & 1.13 & 1.24 \end{pmatrix} . \begin{pmatrix} 1 & -2 & 3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 0.75 & -0.36 & 0.1 \end{pmatrix}
\end{aligned}$$

A solution to the system above is given by $\alpha_1^{3,9} = 0.75$; $\alpha_2^{3,9} = -0.36$; $\alpha_3^{3,9} = 0.1$. *The real numbers* $\alpha_k^{3,9}$ *can now be used to approximate the inclusion-exclusion formula in the correction term as follows:*

$$\begin{aligned}
R_{LN}(S^9, S[10]) &= \sum_{k=1}^{3} \alpha_k^{3,9} \sum_{\substack{O \subseteq L(S,Y) \\ |O|=k}} \phi_{LN}(S^{\min(O)-1}) \cdot \left(2^{|(\bigcap_{j \in O} S[j]) \cap S[10]|} - 1\right) \\
&= 9{,}298\,127\,9 \cdot 10^{26}
\end{aligned}$$

Notice here that the formula contains only $\sum_{i=1}^{3} \binom{9}{i}$ *terms, which is already a significant computation gain in comparison with the* $\sum_{i=1}^{9} \binom{9}{i}$ *terms in the classical approach.*

Finally, the approximated number of distinct subsequences for sequence S *is* $\phi_{LN}(S^{10}) = 2^{|\{a,b,c,d,e,f,g,h,i,j,,k\}|} \cdot \phi_{LN}(S^9) - R_{LN}(S^9, S[10]) = 2{,}524\,495\,6 \cdot 10^{30}$ *which is very close to the above number.*

4.8 Implementation and Experimental Validation

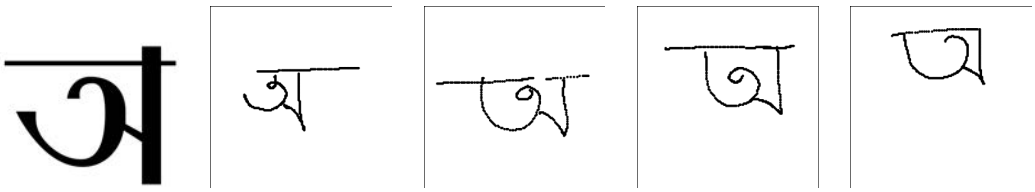
In this section we empirically evaluate our similarity measure on synthetic and real-world datasets. The results reported in this work are a useful contribution with direct practical applications to different discriminative approaches, and in particular kernel methods, because new complex sequence kernels can be devised based on the theoretical results provided in this work. Moreover, the method is quite general in that it can be used (with slight modifications) for a broad spectrum of sequence-based classification or clustering problems. We report an extensive empirical study on synthetic datasets and qualitative experiments with datasets consisting of trajectories

of online handwritten Assamese characters. We give empirical evidence showing that the proposed approximation method works well in practice. Our approach is implemented in both C++ and Java languages. The whole analysis is run over a MacBook Pro with a 2.5GHz Intel Core i5, 4GB of RAM running OS X 10.6.8. A dedicated web page to visualize data sets and interact with the experimental results is available at `http://www.loria.fr/~eegho/acs/`.

4.8.1 Clustering Handwritten Assamese Symbols

Assamese Symbols

This batch of experiments was conducted with handwritten Assamese symbols from the UCI Machine Learning Repository[7]. Assamese is an official language used in the state of Assam in North East of India. It contains 183 symbols including 52 characters, 10 numerals and 121 conjunct consonants. The dataset is collected from 45 users who were instructed to write the symbols on a graphic tablet. The tablet program records the handwriting as a stream of (x, y) coordinate points using the appropriate pen position sensor along with the pen-up and pen-down switching. Figure 4.2 presents a sample of users writing the অ character. In the data set, each symbol for each user is modeled as a sequence of strokes. Each stroke consists of a set of points (x, y) representing the pen position on the tablet (i.e., a stroke represents an itemset). Each sequence representing a symbol was preprocessed by cropping and scaling the resolution of the symbol. Figure 4.3 illustrates the two-steps preprocessing applied on the character অ for a given user. The total number of sequences in the data set is 8235 (45 users drawing each 183 symbols). Figure 4.4 shows the distribution of the number of itemsets (i.e., strokes) for each symbol in the dataset. Our objective is to study the different Assamese handwriting techniques and build user clusters based on their handwriting style.



Original User 1 User 13 User 24 User 44

Figure 4.2: Handwritten character অ

A Comparison Between sim_{ACS} and sim_{LCS}

We compared our similarity measure with the two versions of sim_{LCS}. The similarity between symbols was first computed using sim_{ACS} then with $sim_{LCS_{size}}$ and $sim_{LCS_{length}}$. For a given symbol X, we ranked the other symbols according to their similarity with X. The rank correlation was then analyzed with a scatter plot. Figure 4.5 represents the rank correlation between sim_{ACS} and sim_{LCS} when all the symbols are ranked according to the symbol ব. The non-correlation of the two measures is clear in Figures 4.5a and 4.5b. The remarkable strata displayed in Figure

[7]http://archive.ics.uci.edu/ml/datasets/Online+Handwritten+Assamese+Characters+Dataset

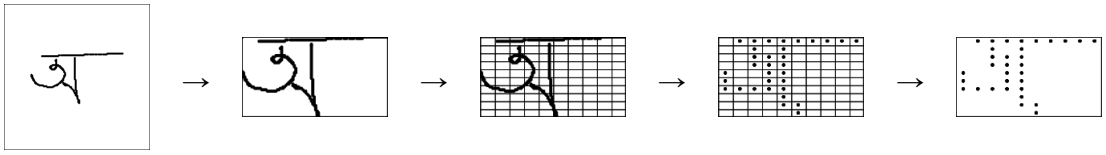


Figure 4.3: Preprocessing steps on Character অ

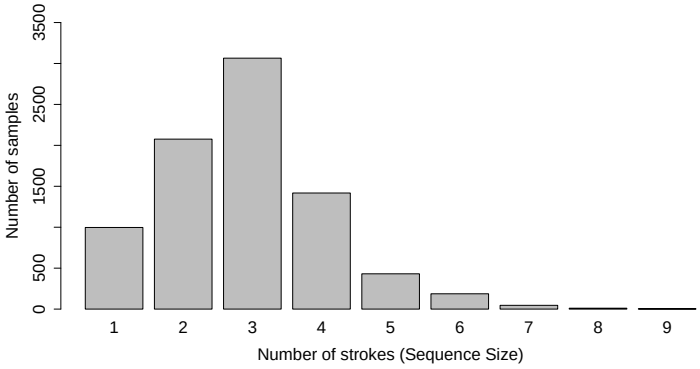


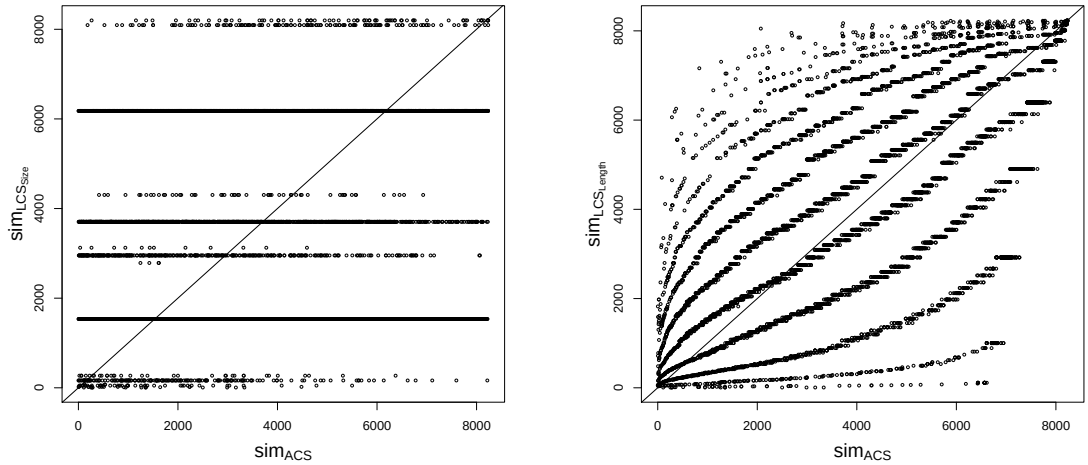
Figure 4.4: Distribution of the number of strokes in the data set

4.5a are the result of the low variance over the similarity measures from $sim_{LCS_{size}}$. For example, character ঝ needs 3 strokes and so $sim_{LCS_{size}}$ between ঝ and all the symbols which needs less than 3 strokes will always be limited to 3 possible values: $\frac{1}{3}$, $\frac{2}{3}$ or 1. The same reasoning applies to symbols with more than 3 strokes as $sim_{LCS_{Size}}$ will have at most 16 possible different values (the maximal number of strokes to write a symbol in the data set is 9). This is a limitation that is not present for the sim_{ACS} measure. To further illustrate this point, notice that for the user 3, $Sim_{LCS_{Size}}$ similarity value between ঝ and ক is $\frac{2}{3}$ and is exactly the same similarity value between ঝ and ল. Yet, sim_{ACS} returns a similarity value for ঝ and ক that is much lager than the similarity between ঝ and ল. The same drawback can be observed with the $sim_{LCS_{length}}$ measure, albeit to a lesser extent, and is illustrated in Figure 4.5b.

Clustering Assamese Handwritten Symbols

The goal of this experiment is to cluster the Assamese characters using sim_{ACS}. To interpret this experiment in a simple manner we will first detail the clustering process for 2 symbols ক্ক and ক্ত over a small sample or 3 randomly selected users, illustrated in Figure 4.6. The whole process over the 45 users and the 8235 symbols is discussed later.

A trivial case such as 6 sequences selected over 3 random users returns contrasting conclusions. Different hierarchical clustering results with sim_{ACS}, $sim_{LCS_{size}}$ and $sim_{LCS_{length}}$ similarity measures are pointed out in Figures 4.7a, 4.7b and 4.7c. Clustering with sim_{ACS} returns good results with a clear partition of the sequences in 2 groups respectively representing the 2 studied



(a) sim_{ACS} versus $sim_{LCS_{Size}}$ (b) sim_{ACS} versus $sim_{LCS_{Length}}$

Figure 4.5: sim_{ACS} and sim_{LCS} rank correlation for character ঝ written by user 3

ক্ষ				ক্ত			
Original	User 4	User 12	User 14	Original	User 4	User 12	User 14

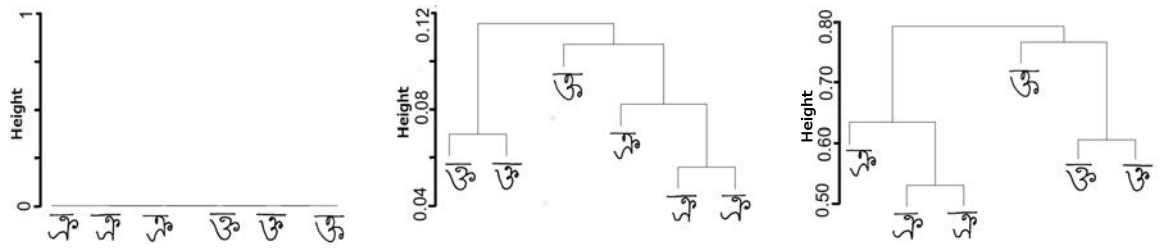
Figure 4.6: How the users 4, 12 and 14 draw the characters ক্ষ and ক্ত

symbols. Clustering with sim_{LCS} on the other side yields more subtle results with $sim_{LCS_{size}}$ clustering all the sequences in the same group with equidistant similarities. $sim_{LCS_{length}}$ gives better results but still incorrectly clusters one of the characters.

	Purity	NMI	RI	F_5
$sim_{LCS_{Size}}$	0.45	0.11	0.5	0.43
$sim_{LCS_{Length}}$	0.63	0.35	0.67	0.52
sim_{ACS}	**0.86**	**0.6**	**0.85**	**0.85**

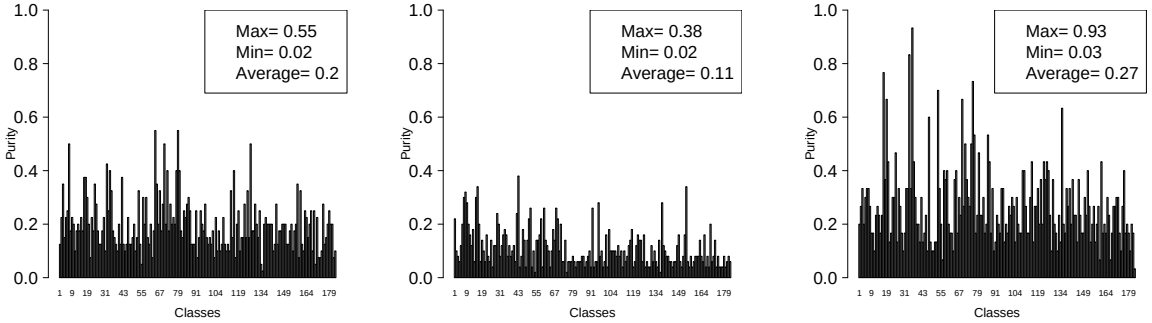
Table 4.3: Clustering evaluation of 90 users draw two characters ক্ষ and ক্ত

In the following experiment, we extended the 2 symbols clustering to the 45 users with a total of 90 sequences. We generated the similarity matrix, applied hierarchical clustering and cut the hierarchical clustering tree at the second level from the top. We computed the Purity, Normalized Mutual Information (NMI), Rand Index (RI) and F measure to evaluate how well the clusters, obtained using the three measures, matched with the original clusters (i.e, the 2 original clusters contain the 45 different handwritten characters ক্ত and ক্ষ). The results are reported in Table 4.3. Notice that sim_{ACS} returns for this experiment more homogeneous clusters.



(a) $sim_{LCS_{Size}}$ (b) $sim_{LCS_{Length}}$ (c) sim_{ACS}

Figure 4.7: Hierarchical clustering results for the users 4, 12 and 14 draw the characters ক and ক



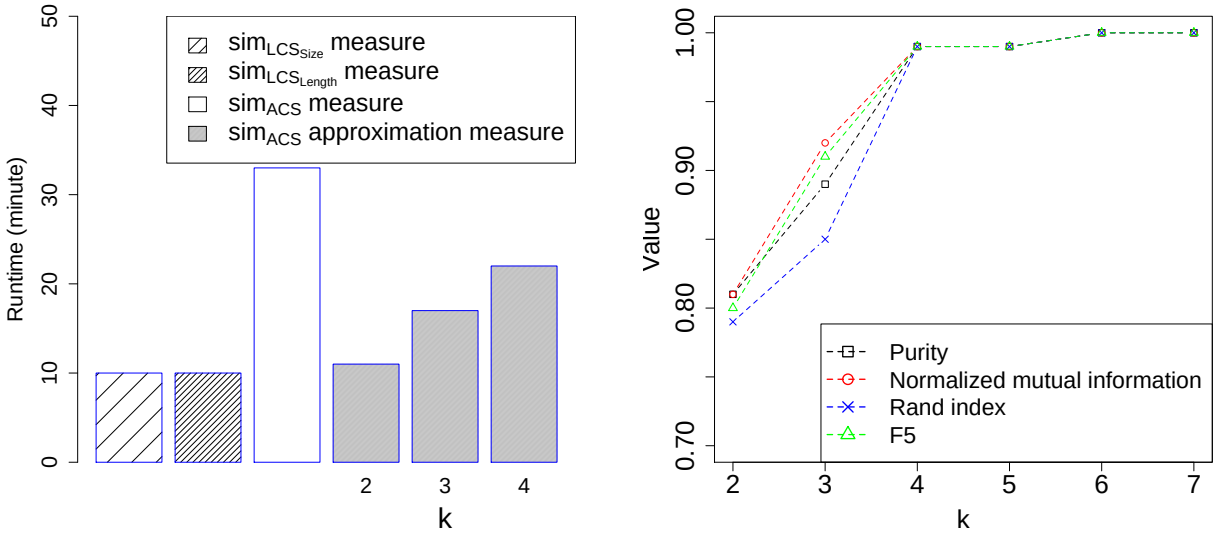
(a) $sim_{LCS_{Size}}$ (b) $sim_{LCS_{Length}}$ (c) sim_{ACS}

Figure 4.8: Purity value for each class of handwritten symbols

In the final experiment, we built a similarity matrix using sim_{ACS} for the 45 users and the 8235 symbols. A hierarchical clustering procedure is also applied. We cut the tree at the 183th level (the number of clusters corresponding to the number of symbols in the alphabet). Our goal is to detect whether we can group all the users who draw the same character in one class. The results with the Purity measure are reported in Figure 4.8. Other detailed results for this experiment are described at the following url: `http://www.loria.fr/~eegho/acs/`.

4.8.2 Linial-Nisan approximation

We built the similarity matrix with several value of $k \in [2, 9]$. Our goal is twofold: (i) compare the clustering quality based on the approximation method and (ii) compare the runtime. We compared the clusters obtained using the Linial-Nisan approximation with the clusters obtained using sim_{ACS}. Figure 4.9b reports the values of each cluster quality measure with several different values of k. We noticed that starting from $k = 4$, the clusters are identical to the clusters built using sim_{ACS}. In terms of runtime, Figure 4.9a reports the different time gains when building the similarity matrix with sim_{ACS}.



(a) Runtime for building the similarity matrix for 8325 symbols

(b) Clusters matching values obtained with the Linial-Nissan approximation

Figure 4.9: Linial-Nisan cluster matching and runtime

4.8.3 Studying Scalability on Synthetic Data Sets

In the following, we study the scalability of our measure computation. We assess the different runtimes with respect to three different parameters:

- The average number of itemsets in a sequence;
- The average number of items in each itemset of a sequence;
- The total number of sequences that are processed through the similarity computation.

We generated our datasets by IBM Quest market-basket synthetic data generator. Table 4.4 describes the synthetic datasets that were generated and used in our experiments.

Figures 4.10 and 4.11 show the evolution of the runtime of 499 500 ($\frac{n\times(n-1)}{2}$) comparisons over 1 000 sequences w.r.t the average number of items in each itemset and the average number of itemsets in each sequence. We run this test on several types of sequences: sequences with itemsets of cardinality 5, 10, 15, 20, 25, 30 (i.e, the number of items in each itemset) and with several lengths: 5, 10, 15, 20 and 25 itemsets (i.e, the number of itemsets in each sequence).

As expected, the plots on Figure 4.10 show that the execution time for calculating the similarity matrix without any approximation increases with the size of the sequences. With the Linial-Nissan approximation, the execution time is greatly reduced as seen on Figure 4.11. The time to compute the similarities with the approximation for 1000 sequences of 25 itemsets is less than 1000 seconds in comparison with more than 150000 seconds (i.e., there is a 150× runtime gain).

Dataset	Itemset Length	Sequences Length	Nb-Sequence
$data_1$	5	5	1000
$data_2$		10	
$data_3$		15	
$data_4$		20	
$data_5$		25	
$data_6$	10	5	1000
$data_7$		10	
$data_8$		15	
$data_9$		20	
$data_{10}$		25	
⋮			
$data_{26}$	30	5	1000
$data_{27}$		10	
$data_{28}$		15	
$data_{29}$		20	
$data_{30}$		25	
$data_{31}$	10	10	2000
$data_{32}$			3000
$data_{33}$			4000
$data_{34}$			5000

Table 4.4: Synthetic Datasets

The next experiment aims at comparing the runtime performances of sim_{LCS} with our proposed similarity measure. Figure 4.12 presents a comparison of the similarity matrix computation runtimes based on sim_{ACS}, sim_{LCS} and sim_{ACS} with the Linial-Nisan approximation. In this experiment, the length of the sequence is fixed to 15 (i.e., 15 items per sequence). The plot illustrates the fact that our similarity measure, along with the Linial-Nissan approximation, takes almost the same runtime as the longest common subsequence computations.

The plot on Figure 4.13 highlights the impact of the parameter k on the runtime for the Linial-Nissan approximation. This figure shows that the calculation time increases by a factor of two on each increment of the value of k.

The final experiment focuses on the similarity matrix computation runtime when the size of the input sequences increases. We run these tests over sequences with 10 itemsets on average and with 10 items for each itemset. For 5 000 sequences (i.e. 12 497 500 similarity comparisons), the execution time for our similarity measure is about 30 minutes (1 832 seconds) and about 11 minutes (660 seconds) for the Linial-Nissan approximation.

Figure 4.14 reports the different runtime observations. These experiments highlight the fact that our measure is efficient in terms of runtime for a large panel of sequences with different varying parameters.

4.9 Conclusion and Perspective

In this chapter, we study the problem of counting all common subsequences between two sequences of itemsets. We present theoretical results and an efficient dynamic programming algorithm (ACS) to count the number of common subsequences between two sequences. This solution allows us to define in a simple and intuitive manner a similarity measure, denoted sim_{ACS}, between two sequences S and T. In addition, we propose an approximation method to speed up the computation for long sequences.

This similarity has been successfully applied for the analysis of handwritten symbols and

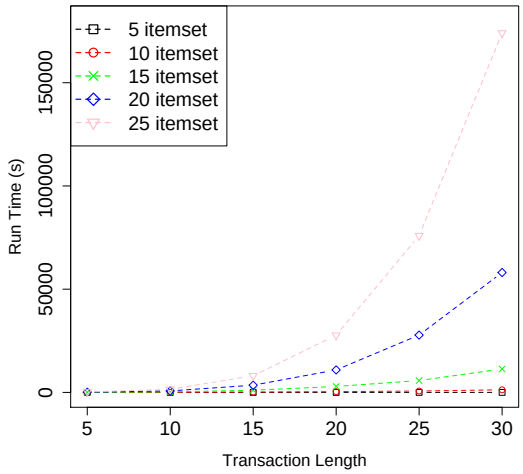


Figure 4.10: 1 000 sequences similarity matrix computation runtime. Various lengths of itemsets

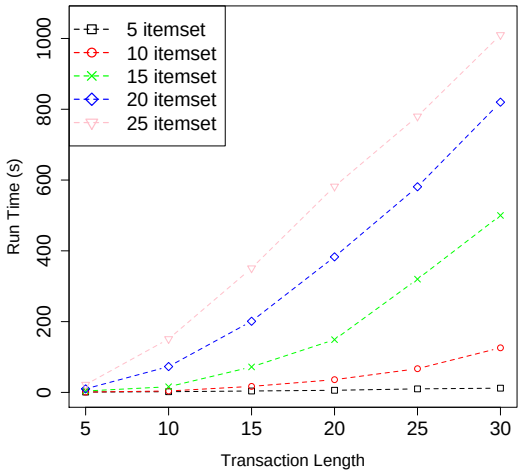


Figure 4.11: 1 000 sequences similarity matrix computation with approximation runtime. Various lengths of itemsets

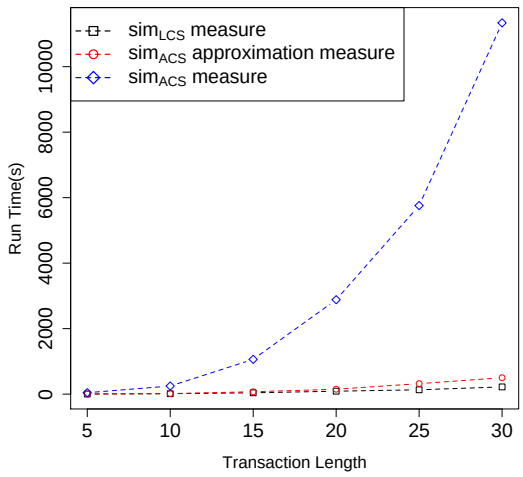


Figure 4.12: Multiple 1 000 sequences similarity matrix computations. Sequences of fixed length: 15 items

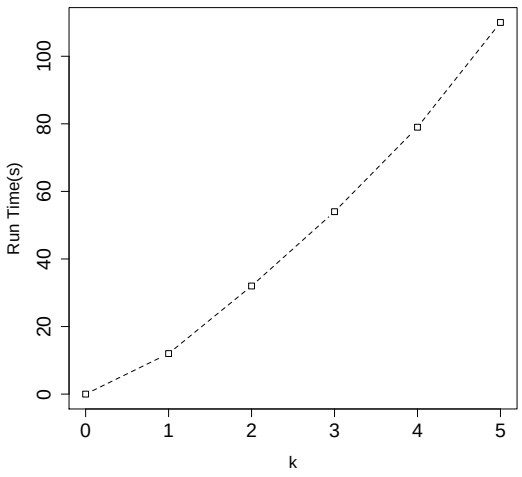


Figure 4.13: 1 000 sequences similarity matrix computation with approximation runtime. Various values of k

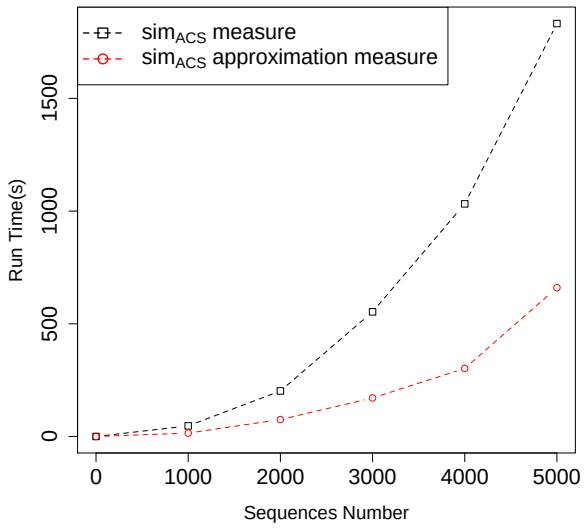


Figure 4.14: Similarity matrix calculation run time w.r.t the number of sequences

synthetic data sets.

Experiments show that sim_{ACS} is more accurate than $sim_{LCS_{Size}}$ and $sim_{LCS_{Length}}$. The clusters obtained by sim_{ACS} are more precise and more complete. Experiments show also that, sim_{ACS} is more sensitive than $sim_{LCS_{Size}}$ and $sim_{LCS_{Length}}$ to sequence modifications when adding a new items or itemsets. sim_{ACS} is more variance than $sim_{LCS_{Size}}$ and $sim_{LCS_{Length}}$. The experiments on synthetic data sets also highlight the fact that our measure is efficient in terms of runtime for a large panel of sequences with different varying parameters.

Several areas and methods as the outliers detection, extraction of sequential patterns under similarity constraint, compression of sequential patterns,information retrieval, visualisation and querying the sequences, etc are possible applications of sim_{ACS}.

We plan to apply the measure on various sequence data sets (more precisely trajectory mining, molecular bioinformatics and text classification). We also intend to compare our measure with kernel methods for sequence classification.

We considered sequences of itemsets. However, it can be extended to multidimensional sequences where a sequence is formed by series of elementary vectors in which each elementary vector is a vector of itemsets and each itemset is an antichain on partially ordered set.

Chapter 5

Detection and Classification of Healthcare Trajectory

Contents

5.1 Introduction 101
5.2 PMSI French healthcare system 102
5.3 Mining Healthcare Trajectories by using *MMISP* 105
5.3.1 Mining healthcare trajectories 105
5.3.2 *MMISP* versus Standard sequential pattern mining method 107
5.3.3 $MMISP$ versus M^3SP 108
5.4 Mining Healthcare Trajectory by using Sequential Pattern Structure 109
5.5 Classification of Healthcare Trajectory by using FCA 112
5.6 Clustering Healthcare Trajectories by using sim_{ACS} 114
5.6.1 Clustering Healthcare Trajectories 114
5.6.2 A Comparison Between sim_{ACS} and sim_{LCS} 115
5.6.3 Linial-Nisan Approximation 117
5.7 Conclusion 118

5.1 Introduction

In this chapter we empirically evaluate our contributions by conducting several experiments on datasets consisting of trajectories of cancer patients extracted from French healthcare organizations that allowed us to evaluate our methods. Our main goals are to provide a good description of healthcare trajectory, better understanding of what are in reality for healthcare trajectory and to group patients according to their trajectories. To achieve these goals, we mine, classify and cluster healthcare patients trajectories and give potential interesting patterns for healthcare specialists. We show the interest of our approaches with the analysis of trajectories of care for cancer diseases using data from PMSI[8], the French casemix information system.

To mine healthcare patients trajectories, we apply *MMISP* on a set of patients suffering from lung cancer and living in the Lorraine region. We present that the patterns extracted by *MMSIP* help healthcare managers and deciders in planning and organizing healthcare resources

[8]Programme de Médicalisation des Systèmes d'Information

at a regional level. Then, we compare our method, *MMISP*, with two existing methods for mining sequential patterns such as *CloSpan* proposed by Yan et al. [2003] and M^3SP proposed by Plantevit et al. [2010].

The sequential pattern structure is tested on the same healthcare information system PMSI. Interesting patterns are extracted and then interpreted, showing the feasibility and usefulness of the sequential pattern structure with the projection.

To classify the healthcare trajectories mined with *MMISP*, we use FCA lattice-based classification method. This experiment provides two advantages, first, it helps in grouping patient trajectories sharing some attributes and producing their graphical visualizations. The second benefit of using FCA is that it is able to characterize groups of patients showing several trajectories. Finally,by pruning concept lattices with the help of support and stability measures to filter out only interesting group of patient trajectories.

Another way of grouping patients according to their trajectories is to cluster the healthcare trajectories by studying the similarity between them. We use our similarity measure, sim_{ACS}, to build a similarity matrix between patient trajectories. A hierarchical clustering procedure was then applied to divide patient trajectories into several clusters. We compare also sim_{ACS} with the two sim_{LCS} similarity measures (i.e., using either sequence length or size). Then, we highlight the fact that approximating our similarity measure still yields good and competitive conclusions, with fast computation times. A dedicated web page to visualize our dataset and interact with the experimental results is available at `http://www.loria.fr/~eegho/acs/`.

The remaining of this chapter is organized as follows. Section 5.2 presents PMSI, the french casemix information system. In section 5.3, we present some of the results we achieved with *MMSIP*. Section 5.4 presents the sequential pattern structure to deal with healthcare trajectories. Section 5.5 shows how we classify patient trajectories by using FCA. The section 5.6 shows how we use sim_{ACS} to divide patient trajectories into different clusters.

5.2 PMSI French healthcare system

In France, the *"Programme de Médicalisation des Systèmes d'Information"* (PMSI) is a nationwide information system, derived from the Diagnosis Related Groups (DRG) system Fetter et al. [1980]. The PMSI system has two important advantages for the analysis of trajectories of care. Each sector of activity (acute, post-acute, psychiatric care) is covered by an information system common to the whole French population. Second, since the introduction of an anonymised identifier in 2001, it allows the linkage of all hospitalizations of a same patient across time, space and sectors of activity.

In this system, each hospital stay leads to the collection of a standardized set of administrative (age of patient, length of stay ...) and medical data (primary diagnosis, comorbidities, medical and surgical procedures). Figure 5.1 shows the entity relationship model of the data in PMSI system. Its structure can be described (and voluntarily simplified) as follows:

- Entities (attributes):
  - Patient: (id, gender, date of birth ...)
  - Hospitalization: (id, ...)
  - Associated Diagnoses: (ICD10 code)
  - Medical Procedures: (CCAM code)
- Relationships

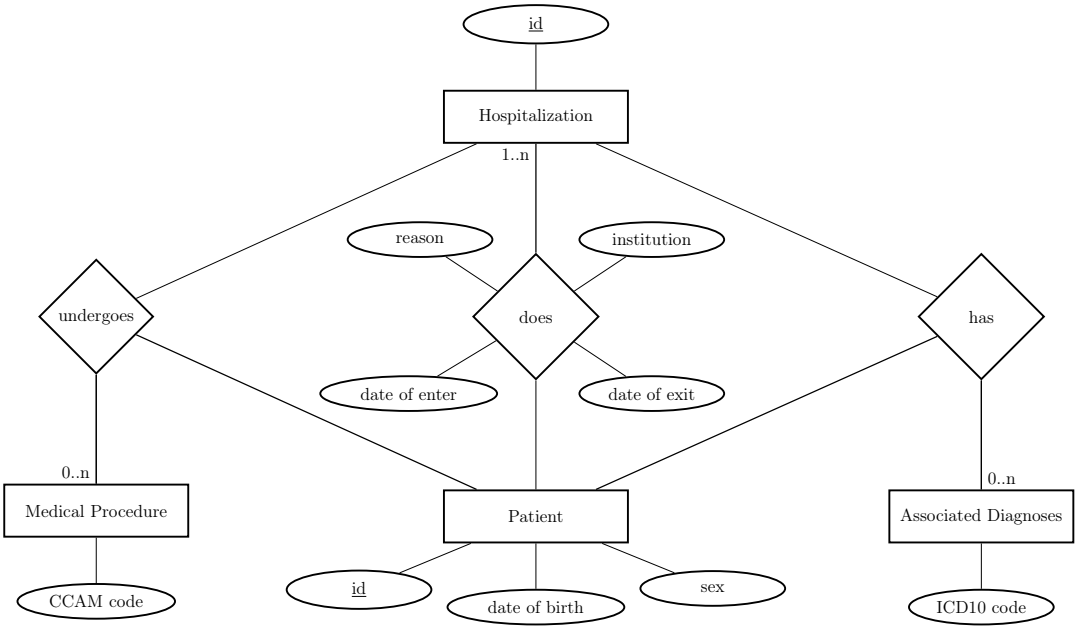


Figure 5.1: Entity relationship model of the data in the PMSI system

- A patient does one or more hospitalizations at a specific date and in a healthcare institution for a specific reason (principal diagnoses).
- During the hospitalization, a patient may undergo several medical procedures.
- During the hospitalization, a patient may have several associated diagnoses.

The collection of data is done with a minimum recordset using controlled vocabularies and classifications. Diagnoses are coded with the 10th International Classification of Diseases (ICD10). Based on this coding, diagnoses could be described at 5 levels of granularity: root, chapter, block, 3-character, 4-character, terminal nodes. Figure 5.2 depicts chapters such as $Neoplasms$ with specializations: block $C30 - C39$ is a malignant neoplasms of respiratory system, block $C50 - C50$ which is a malignant neoplasms of breast etc. The block $C30 - C39$ has specializations: malignant neoplasm of larynx (code: $C32$), malignant neoplasm of bronchus and lung (code: $C34$), etc. $C34$ (Lung cancer) has specializations: $C340$ is a cancer of the main bronchus, $C341$ is a cancer of upper lobe etc.

The *medical procedures* are coded with the french nomenclature "Classification Commune des Actes Médicaux (CCAM)". They are represented by a taxonomy with 5 levels of granularity. For example, $ZBQK002$ is a chest radiography for respiratory system, which is one of the procedures applied for respiratory diseases. These procedures are a part of the respiratory procedure which in turn is included in the medical procedure.

The *healthcare institution* was associated with a geographical taxonomy of 4 levels, first level refers to the root (France) and second, third and fourth level correspond to administrative region, administrative department and hospital respectively. Figure 5.3 illustrates University Hospital of Nancy (code: 540002078) as a hospital in Meurthe et Moselle, which is a department in Lorraine in the Region of France. The healthcare institution can also be represented based on its type with a taxonomy of 3 levels. The first level refers to the root, while the second and third level correspond to type and the healthcare institution respectively. Figure 5.4 illustrates University Hospital of Nancy (code: 540002078) as a university hospital (CHU), which is a type

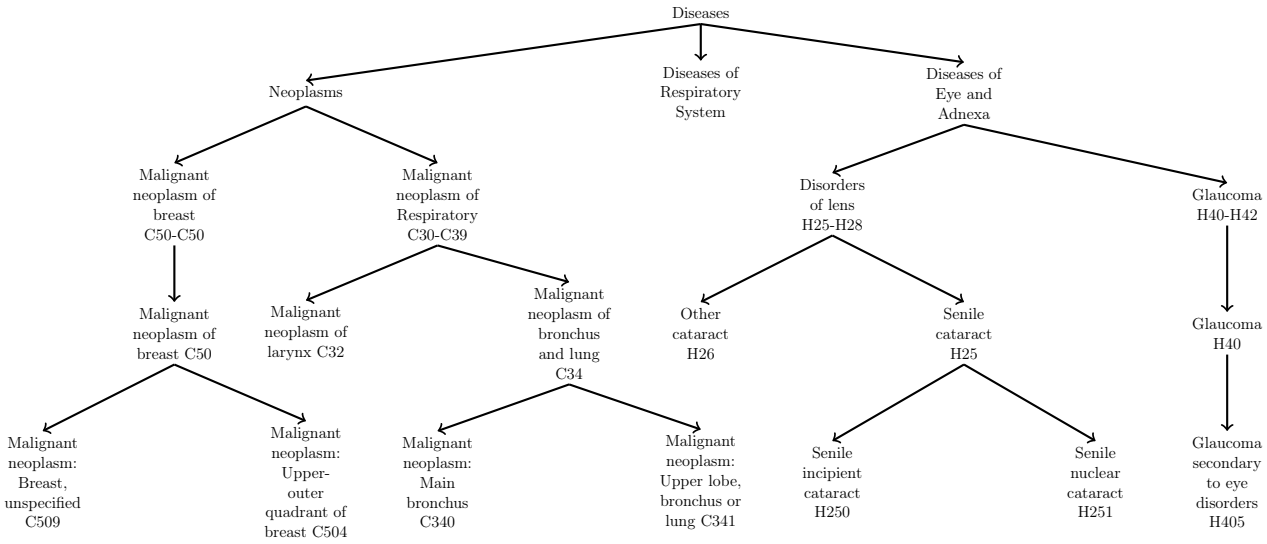


Figure 5.2: A disease taxonomy

of a healthcare institution.

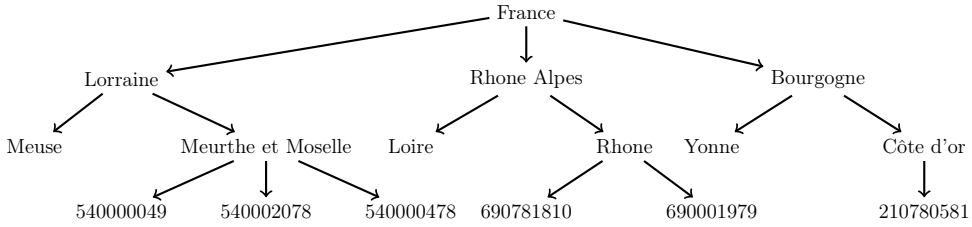


Figure 5.3: A geographical taxonomy of the healthcare institution

Most of existing patient classification systems, the PMSI focuses on single contacts and was not designed to categorize a care process spanning several hospitalizations. For chronic conditions, it is of much interest to summarize the information contained in a sequence of several hospitalisations and produce meaningful categories that will be relevant for subsequent analysis. Such a goal cannot be reached without a recomposition and a better understanding of the so called healthcare trajectories. As chronic patients can have multiple diagnoses and multiple treatments recorded in several different facilities, mining and classifying healthcare trajectories are difficult and time consuming tasks.

The sequence of hospitalization for one patient defines his care trajectory. One hospitalization is made up of various aspects such as healthcare institutions with patient's diagnosis and patient's medical procedures. The analysis of healthcare trajectory can take in account any combination of these aspects, yielding different point of views. Meanwhile, data mining methods and more specially our approaches may support the experts in the categorization and analysis of trajectories of care. In the next section, we use a *MMISP* as a method to mine healthcare patients trajectories and give potential interesting patterns for healthcare specialists.

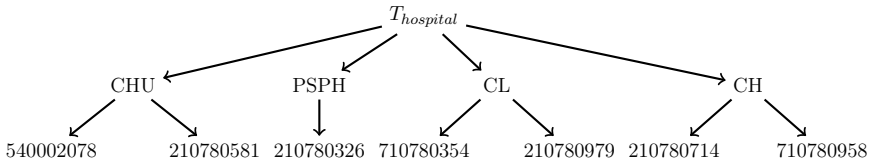


Figure 5.4: Healthcare institutions taxonomy depending on their types

5.3 Mining Healthcare Trajectories by using *MMISP*

5.3.1 Mining healthcare trajectories

This section describes the results obtained after applying MMISP on a set of 100 patients suffering from lung cancer who live in the Lorraine region, of Eastern France. We reconstituted the sequence of hospitalizations of patients who have treatment between 2006 and 2010. Each hospitalization in a trajectory was characterized by elementary vector including three elements: *hospital*, *principal diagnosis* and *medical procedures* delivered during the stay. Each element in the hospitalization can be represented at different levels of granularity, by using a taxonomy. We use a geographical taxonomy for the healthcare institution (see Figure 5.3), the ICD10 classification for the diagnosis (see Figure 5.2), while the medical procedures are classified based on CCAM[9] code. Figure 5.5 shows the distribution of the length of sequences of care in our dataset.

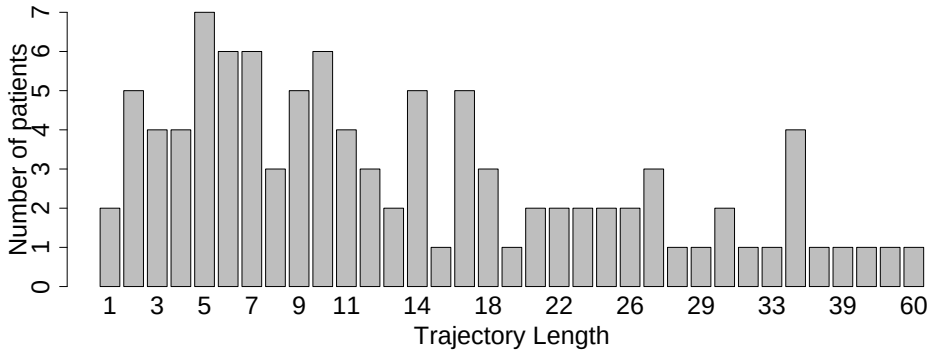


Figure 5.5: Distribution of the length of patient trajectories

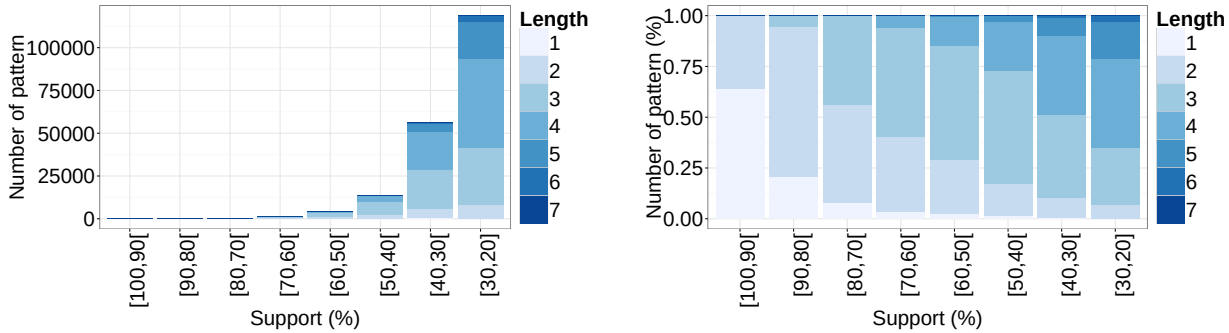
Table 5.1 shows an example of care trajectories for 3 patients. For example, $Patient_1$ has two hospitalizations, the first was in the University Hospital of Nancy (code: 540002078) for lung cancer (code: $C341$) where he underwent a chest radiography (code: $ZBQK$). Then, he was hospitalized in a private clinic in Metz (code: 570023630), for a chemotherapy session (code: $Z51$) where he had a chest radiography and pneumonectomy (code: $GFFA$).

In this experiment, the support value is set to 20 patients (i.e. $\sigma = 20\%$). *MMISP* generates *194 650* different frequent trajectories. Figure 5.6.a shows the number of discovered patterns at different thresholds according to their length. With support between 30% and 20%, the high

[9]Classification Commune des Actes Médicaux

Patients	Trajectories
$Patient_1$	$\langle(540002078, C341, \{ZBQK\}), (570023630, Z51, \{ZBQK, GFFA\})\rangle$
$Patient_2$	$\langle(100000017, C770, \{ZBQK\}), (210780581, C770, \{ZZQK, YYYY\}), \ldots\rangle$
$Patient_3$	$\langle(210780110, H259, \{YYYY\}), (210780110, H259, \{ZZQK\}), \ldots\rangle$

Table 5.1: Care trajectories of 3 patients



(a) Number of sequential patterns by support and length.

(b) Percent of sequential patterns by support and length.

Figure 5.6: A cumulative description of sequential patterns by support and length

number of length 3 and 4 patterns is explained by a combinatorial effect resulting from a high number of sequences of length 5-11 in the database. These frequent sequences correspond to the patients who underwent chemotherapy and usually had around 3 and 6 stays for 1 cycle. With support threshold equals to 100%, there is only one pattern $\langle(\top_{hospital}, C34, \{ZBQK\})\rangle$ which shows that 100 % of the patients had a Lung cancer (code: $C34$) and underwent chest X-Ray (code: $ZBQK$) during a visit.

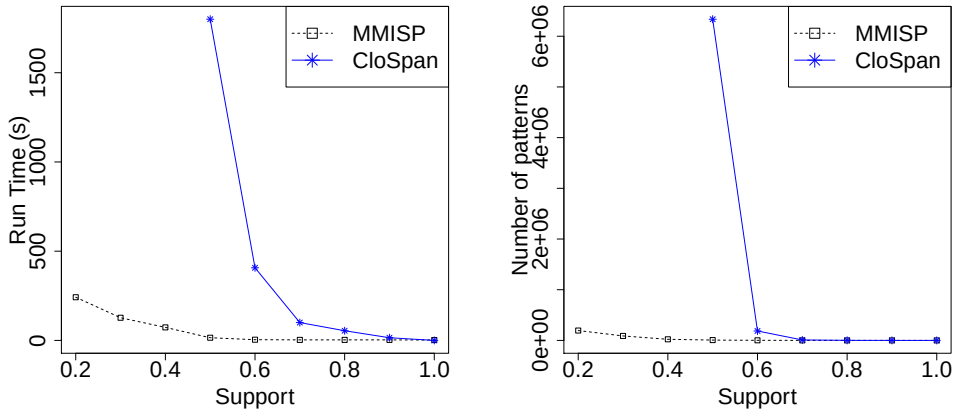
Table 5.2 shows the items appearing in *principal diagnosis* dimension of patterns for which support is over 27%. It can be noticed that the ICD10 tree has been mined at different levels. In the neoplasm branch, the most specific observed item is of depth 3, *" malignant neoplasm of bronchus and lung"*. In the branch of *"factors influencing health status and contact with health services"*, items of depth 4 (*"chemotherapy session for neoplasm"*) have been extracted. Children of *"Malignant neoplasm of bronchus and lung"* are not frequent enough to be extracted, but *"chemotherapy session"* appears in a sufficient proportion of trajectories to be seen. Such results cannot be obtained by representing items at an arbitrary pre-determined level.

The mds-patterns can be analyzed per se. For example, the pattern $\langle(Lorraine, C34, \{ZBQK, GFFA\})\rangle$ shows that 93% of patients had pneumonectomy (code: GFFA) and chest X-Ray (code: ZBQK) for a lung cancer (code: C34) in any hospital in Lorraine Region in France. The mds-pattern $\langle(Lorraine, \top_{Diag}, \{06.01\}), (Lorraine, C34, \{ZBQK,\ GFFA\}), (Lorraine, \top_{Diag}, \{06.01.03\})\rangle$ shows that 59% of patients had three hospitalizations where in the first one they started their treatment by underdoing diagnostic test of the respiratory system (code: 06.01) then having pneumonectomy and chest X-Ray for a lung cancer and a subsequent stay in the Lorraine Region for complementary treatments and follow-up.

This kind of information helps healthcare managers and deciders in planning and organizing healthcare resources at a regional level. Besides, sequential patterns can be seen as a condensed representation of care trajectories. As such, patterns can be reused as new variables to distinguish subgroups of patients in subsequent analysis.

ICD10 level – Diagnosis Taxonomy
0– Root
1– Neoplasms
2– Malignant neoplasms of respiratory and intrathoracic organs (C30–C39)
3– Malignant neoplasm of bronchus and lung (C34)
1– Factors influencing health status and contact with health services
2– Persons encountering health services for specific procedures and health care (Z40-Z54)
3– Other medical care (Z51)
4– Chemotherapy session for neoplasm (Z511)

Table 5.2: Items extracted in the Principal Diagnosis dimension, (minimal support equals to 27%)



(a) Runtime sequences over frequency threshold

(b) Number patterns over frequency threshold

Figure 5.7: *MMISP* versus *CloSpan*

5.3.2 *MMISP* versus Standard sequential pattern mining method

In this section, we compare *MMISP* with a standard sequential pattern mining method such as *CloSpan* Yan et al. [2003]. All standard sequential pattern mining algorithms require that the dataset to be mined is composed of pairs of the form (id, seq), where id is a sequence identifier and seq is a sequence of itemsets. We transfer each sequence s_i in $\mathcal{MS}_{DB}$ with an *extended-sequence* $\widehat{s}_i$. Each elementary vector of a sequence s_i is transformed into a single itemset by replacing its elements with all its ancestors. For example, the elementary vector $(uh_p, ca_1, \{mp_{111}, mp_{221}\})$ would be replaced with $\{\top_h, uh, uh_p, \top_d, ca, ca_1, \top_{mp}, mp_1, mp_2, mp_{11}, mp_{22}, mp_{111}, mp_{221}\}$ as $uh_h \leqslant uh \leqslant \top_h$, $ca_1 \leqslant ca \leqslant \top_d$, $mp_{111} \leqslant mp_{11} \leqslant mp_1 \leqslant \top_{mp}$ and $mp_{221} \leqslant mp_{22} \leqslant mp_2 \leqslant \top_{mp}$. Then, we apply *CloSpan* as a sequential pattern mining algorithm on the *extended-sequential* database. This way of managing hierarchies has been used in *GSP* which is proposed by Srikant and Agrawal [1996].

Our main goal is to evaluate the quality of the frequent trajectories mined with *MMISP* and its performance compared to naive approach using *CloSpan*. We firstly transform the 100 patients trajectories, then we apply the two approaches using minimum supports threshold ranging from

100% to 20%. Figure 5.7 reports the execution time and the number of frequent trajectories according to different values of support threshold for both *CloSpan* and *MMISP*.

Actually, *CloSpan* cannot finish its calculations for support threshold less than 50% because the transformation increases number of items in itemset and generates a large number of density similar sequences. Whereas, *MMISP* runs in acceptable time for support as low as 20%.

MMISP is able to extract condensed frequent trajectories w.r.t. the ones mined by *CloSpan*. For example, the trajectory $\langle\{\top_{Hospital}, Lorrine, \top_{Diseases}, T_{Procedure}, Z, Z40-Z54\}, \{\top_{Hospital}, Lorrine, T_{Diseases}, Neoplasms, C30{-}C39, C34, T_{Procedure}, Respiratory\ procedure, Respiratory\ diseases\ procedure, ZBQK, Anatomic\ Pulmonary\ Resection, GFFA\}, \{\top_{Hospital}, Lorrine, \top_{Diseases}, T_{Procedure}, Z, Z40-Z54, Z51\}\rangle$ generated by *CloSpan* contains redundant information as a hospitalization containing $Z51$ will also contain $T_{Procedure}$, Z and $Z40-Z54$. *MMISP* is not affected by this pattern because it extracts just the most specific frequent elementary vector in the first step of the algorithm.

CloSpan extracts all the frequent trajectories (i.e., the general and specific ones) while *MMISP* generates just the more specific ones. For example, if $\langle(\top_{Hospital}, C_{34}, \{ZBQK\})\rangle$ is a frequent trajectory. *MMISP* does not extract the trajectories which are more general like $\langle(\top_{Hospital}, T_{Diseases}, \{ZBQK\})\rangle$ while *CloSpan* extracts both the general and specific ones. Figure 5.7 shows the difference between the number of frequent trajectories extracted by *CloSpan* and *MMISP*. For example, with a support threshold of 70 %, *MMISP* extracts *565 frequent trajectories* while *CloSpan* extracts *6 335 683 frequent trajectories*.

Finally, we may conclude that:

- *MMISP* is more efficient than *CloSpan* over *extended-sequential* database with low support threshold.
- The trajectories extracted by *CloSpan* require post processing while this is not the case with *MMISP*.
- *MMISP* extracts just the most specific frequent trajectories while *CloSpan* extracts both general and specific ones. This means that *CloSpan* extracts a huge number of patterns. Analyzing all of them is not an easy task for healthcare managers and decision makers.

5.3.3 $MMISP$ versus M^3SP

Another experiment is carried out for comparing M^3SP with *MMISP*. Our main goal is to evaluate the effectiveness of sequential patterns mined by *MMISP* compared to the ones extracted by M^3SP. For this purpose, we applied M^3SP with hospital, diagnosis and medical procedures as analysis dimensions. The support value is set to 20 patients (i.e. $\sigma = 20\%$). Table 5.3 reports an example of the extracted patterns with M^3SP and *MMISP*.

Firstly, we observe that *MMISP* is able to extract condensed trajectories w.r.t. the ones mined by M^3SP. For example, *48 trajectories*, *Pattern #1*,..., *Pattern #48*, generated by M^3SP are summarized by *3 ones*, *Pattern #50*, *Pattern #51* and *Pattern #52*, extracted by *MMISP* (see Table 5.3). This shows that the rigid structure of multidimensional item assumed by M^3SP limits the expressivity of the results.

Besides that, in M^3SP, several dimensions can be repeated at the same hospitalization. For example, in M^3SP, *Pattern #48* represents one hospitalization including 9 multidimensional items. Each multidimensional item is associated with the same value of hospital and diagnosis (540002078 and C341) and different values of medical procedures. In *MMISP*, *Pattern #50* (extracted by *MMISP*) represents the same trajectory as *Pattern #48*. *Pattern #50* has one

Methods	id	Trajectory Patterns
M^3SP	1	$\langle\{(540002078, C341, GFFA)(540002078, C341, ZZQK)\}\rangle$
	2	$\langle\{(540002078, C341, DEQP)(540002078, C341, GFFA)(540002078, C341, ZZQK)\}\rangle$
		$\vdots$
	48	$\langle\{(540002078, C341, ZBQK)(540002078, C341, DEQP)(540002078, C341, GFFA)$ $(540002078, C341, GLLD)(540002078, C341, GELD)(540002078, C341, ZZQK)$ $(540002078, C341, GELE)(540002078, C341, FCFA)(540002078, C341, , AGLB)\}\rangle$
	49	$\langle(Lorraine, Diseases\ of\ the\ respiratory, ACQH)\rangle$
$MMISP$	50	$\langle\{(540002078, C341, \{ZBQK, DEQP, GFFA, GLLD, GELD, ZZQK, GELE, FCFA, AGLB\}\rangle$
	51	$\langle(540002078, C341, \{DEQP, GELD, GELE, ZZQK, AGLB, GLLD, GFFA\})\rangle$
	52	$\langle(540002078, C341, \{ZBQK, DEQP, GELD, GELE, ZZQK, GLLD, GFFA\})\rangle$
	53	$\langle(Lor, \top_{Diseases}, \{GEQE, ACQH, ZCQH\})\rangle$

Table 5.3: Some patterns obtained by M^3SP and MMISP.

elementary vector with three elements: hospital 540002078, diagnosis $C341$ and a set of medical procedure $\{ZBQK, DEQP, GFFA, GLLD, GELD,\ ZZQK, GELE, FCFA,\ AGLB\}$. *Pattern #50* is much more compact and informative than *Pattern #48*.

Given a minsup threshold, *MMISP* extracts sequential patterns that are not found by M^3SP. For instance, *Pattern #53* extracted by *MMISP* is not found by M^3SP. This is due to the fact that *MMISP* extracts new frequent hospitalizations not extracted by M^3SP. For instance e*=(Lor,* $\top_{Diseases}$*, {GEQE, ACQH, ZCQH})* and e'*=(Lor,Diseases of the Respiratory, {ACQH})* are extracted by *MMISP*. As e and e' are frequent and not comparable (i.e. $e \not\leqslant e'$ and $e' \not\leqslant e$), M^3SP extracts only *(Lor, Diseases of the Respiratory, ACQH)* and not *(Lor,* $\top_{Diseases}$*, ACQH)* as *(Lor, Diseases of the respiratory, ACQH)* is more specific than *(Lor,* $\top_{Diseases}$*, ACQH)*.

From a quantitative point of view, *MMISP* extracts *419 frequent hospitalizations* with *194 650 frequent trajectories* while M^3SP extracts *102 multidimensional items* with *1 242 frequent trajectories*. The execution time of M^3SP is about 82 *seconds* while *MMISP* takes about 242 *seconds*.

Finally, we may conclude that:

- Several frequent trajectories generated by M^3SP can be summarized by only one mined by *MMISP*.
- Several multidimensional items generated by M^3SP can be summarized by only one elementary vector in *MMISP*.
- One elementary vector in *MMISP* represents one hospitalization in the trajectory while one multidimensional item in M^3SP represents only a part of hospitalization in the trajectory.
- Some frequent trajectories can be extracted by *MMISP* while they can not be extracted by M^3SP.

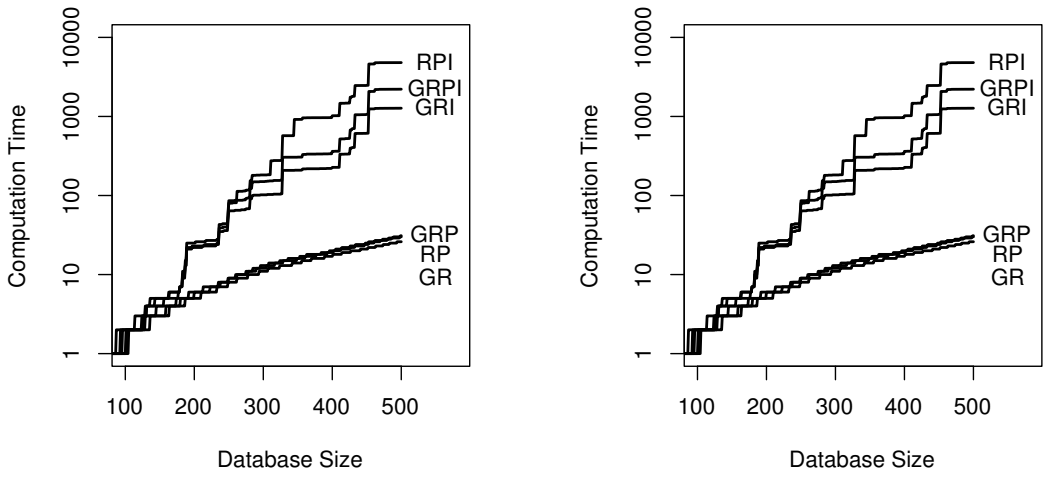
5.4 Mining Healthcare Trajectory by using Sequential Pattern Structure

Mining healthcare trajectories by using *MMISP* algothim generates a large number of frequent sequential patterns while few of them are truly relevant. To echo this challenge, we proposed *a sequential pattern structure* as another solution to mine healthcare trajectories which contains

only consecutively ordered hospitalizations, and then we use *a stability index* to select the interesting ones. We evaluate our proposition on a French national information system PMSI. Our dataset contains 500 patients suffering from *lung cancer*, who live in the Lorraine region, of eastern France. The hospitalization of patient is presented by vector of three elements: (i) healthcare institution, (ii) reason for the hospitalization and (iii) set of medical procedures. The healthcare institution was associated with a geographical taxonomy (see Figure 5.3), while hospitalization reasons and medical procedures are simple sets without associating them with any hierarchical structure. The experiments reveal interesting (from a medical point of view) and useful frequent trajectories, and show the feasibility and the efficiency of our approach. In this experiment, we substituted the same consequent hospitalizations in patient trajectory by the number of repetitions. With this substitution, we have shorter and more readable trajectory. For example, the following trajectory $\langle(chu_{Paris}, cancer, \{mp_1, mp_2\}), (ch_{Lyon}, chemo), (ch_{Lyon}, chemo)\rangle$ should be transformed into two hospitalizations where the first hospitalization repeats once and the second twice:

$$\langle(chu_{Paris}, cancer, \{mp_1, mp_2\}) \times 1, (ch_{Lyon}, chemo) \times 2\rangle$$

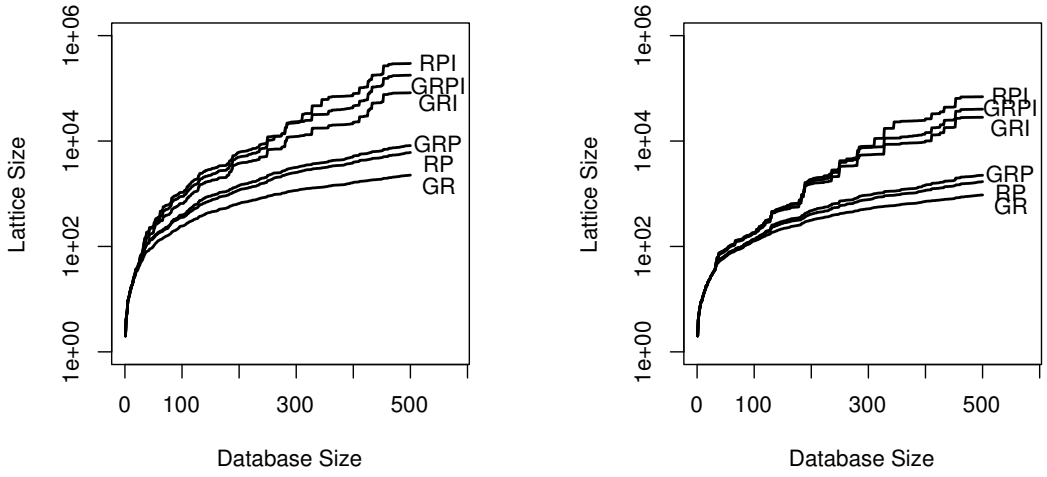
With 500 patient trajectories, the computation of the whole lattice is infeasible. We are not interested in all possible frequent trajectories, but rather in trajectories which answer medical analysis questions. An expert may know the minimal size of trajectories that he is interested in, i.e. setting the MLP projection. We use the MLP projection of length 2 and 3 and take into account that most of the patients has at least 2 hospitalizations in the trajectory.



(a) MLP projection, $\ell = 2$ (b) MLP projection, $\ell = 3$

Figure 5.8: Computational time for different projections

Figure 5.8 shows computational times for different projections as a function of dataset size. Figure 5.8.a shows different alphabet projections for MLP projection with $\ell = 2$, while Figure 5.8.b for MLP with $\ell = 3$. Every alphabet projection is given by the name of fields, that are considered within the projection: `G` corresponds to hospital geo-location, `R` is the reason for a hospitalization, `P` is medical procedures and `I` is repetition interval, i.e. the number of consequent hospitalizations with the same reason. We can see from this figures that MLP allows one to save some computational resources with increasing of ℓ. This difference between $\ell = 2$ and $\ell = 3$ is not that big but it is still useful. A bigger variation can be remarked for alphabet projections.



(a) MLP projection, $\ell = 2$ (b) MLP projection, $\ell = 3$

Figure 5.9: Lattice size for different projections

For example, computation of the `RPI` projection takes 100 times more resources than any from `GRP, RP, GR, GRP`.

The same dependency we can see in Figure 5.9, where the number of concepts for every projection is shown. Consequently, it is important for an expert to provide a strict projection that allows him to answer his questions in order to save computational time and memory.

Table 5.4 shows some interesting concept intents with the corresponding support and ranking w.r.t. to concept stability. For example the concept #1 is obtained under the projection GR (i.e., we consider only hospital and reason), with the intent $\langle[Lorraine, Lung\ Cancer]\rangle$, this concept is the most stable concept in the lattice for the given projection, and the size of the concept extent is 287 patients.

#	Projection	Intent	Stab. Rank	Support
1	GR	$\langle[Lorraine, Lung\ Cancer]\rangle$	1	287
2	$GR2$	$\langle[Lorraine, Respiratory\ Disease]; [CHU_{Nancy}, Lung\ Cancer]\rangle$	26	22
3	$GR3$	$\langle[Lorraine, Chemotherapy] \times 4\rangle$	1	176
4	$RPI3$	$\langle[Preparation\ for\ Chemotherapy, \{Lung\ Radiography\}]; [Chemotherapy] \times [3, 4]]\rangle$	5	36

Table 5.4: Interesting concepts, for different projections.

One of the questions that the analyst would like to address here is *"Where do patients stay (i.e. hospital location) during their treatment, and for which reason ?"*. To answer this expert question, we consider only healthcare institutions and reason fields, requiring both to "have" some information and we use the MLP projection of length 2 and 3 (i.e. projections $GR2$ and $GR3$). Nearly all frequent trajectories show that patients usually are treated in the same region. However, *pattern #2* obtained under $GR2$ projection shows that, *"22 patients were first admitted to any healthcare institution in Lorraine region for a problem related to the respiratory system and then they were detected with the lung cancer in University Hospital of Nancy."*

Another interesting question is *"What are the sequential relations between hospitalization reasons and corresponding procedures?"*. To answer this question, we are not interested in healthcare institutions. Thus, any alphabet element is projected by substituting healthcare institution fields by the '*' hospital. As hospitalization reason is important in each hospitalization, any alphabet element without the hospitalization reason is of no use and is projected to the bottom element

Dimensions used	Number of patterns
`Healthcare Institutions`	1529
`Principal Diagnosis, Healthcare Institution`	4051
`All Associated Diagnoses`	50549
`Principal Diagnosis, Healthcare Institution, Medical Procedures`	321713

Table 5.5: Number of patterns generated by MMISP (`minsup`=5%)

$\perp_E$ of the alphabet. Such projections are called $RPI2$ or $RPI3$, meaning that we consider the fields "Reason" and "Procedures", while the reason should not be empty an the MLP parameter is 2 or 3 *Pattern #4* trivially states that, *"36 patients with lung cancer are hospitalized once for the preparation of chemotherapy and during this hospitalization they undergo lung radiography. Afterwards, they are hospitalized between 3 and 4 times for chemotherapy."*

5.5 Classification of Healthcare Trajectory by using FCA

Mining healthcare trajectories either by *MMISP* or by *sequential pattern structure* generates a huge number of patterns. The number of extracted patterns dramatically grows with the size of the database, depending on the number of sequences, number of aspects in the hospitalization and size of taxonomies. For experimenting this dramatical growth of the number of extracted patterns, we apply *MMISP* method on 486 patients suffering from lung cancer and living in the French region of Burgundy. In our experiment, one hospitalization could be made up of various combinations of spects such as healthcare institutions or healthcare institutions with patient's diagnosis, etc. Table 5.5 shows the number of patterns extracted from 486 patients trajectories. We can notice a dramatic growth in the number of extracted patterns when we increase the number of elements which are presented in the hospitalization. We use a FCA as a classification method to the healthcare trajectories extracted by *MMISP* for interpretation and selection of interesting ones and producing their graphical visualizations

We illustrate this approach with trajectories representing the sequences of healthcare institutions that are frequent in the patients set. These trajectories are generated by applying *MMISP* method on patient trajectories. Their hospitalizations are made by considering only the healthcare institutions and associating them with taxonomy of two levels of granularity (see Figure 5.4).

We build a formal context with 486 patients and 1529 frequent trajectories. The resulting lattice has 10145 concepts organized on 48 different levels. Figure 5.10 shows the upper part of the lattice. Concepts intents are sets of one or more sequential patterns. From the lowest center concept, we can see that 37 patients support 2 sequential patterns:

- at least one hospitalization in the hospital CHU_{Lyon}
- they had at least 2 hospitalization, for simplicity, $\langle(\top_{hospital}) \times 2\rangle$ is the contraction of $\langle(\top_{hospital})(\top_{hospital})\rangle$.

The intent of top concept is $\langle(\top_{hospital})\rangle$, because all patients have at least one hospitalization during their treatment. The intent of co-atoms (i.e. immediate descendant of top) is always a sequence of length one, holding items of high level of granularity.

Filtering concepts can be achieved using both support and stability. In order to highlight the interesting properties of stability, we try to answer the question *"Is there a number of hospitalizations that characterizes care trajectories for lung cancer?"*. A basic scheme in lung cancer

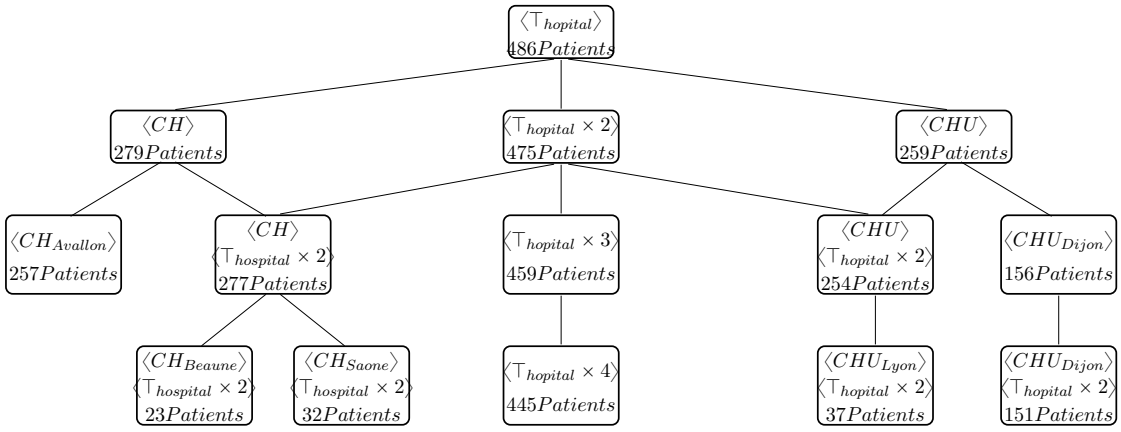


Figure 5.10: Lattice of sequences of healthcare institutions

treatment consists generally in a sequence of 4 chemotherapy sessions possibly following a surgical operation. Due to noise in data or variability in practices, we may observe sequences of 4, 5, 6 or more stays in the PMSI database. Mining such data with an *a priori* fixed support threshold may not discover the most interesting patterns. If the threshold is too high, we simply miss the good pattern. If it is too low, similar patterns, differing only in length, with close values of support can be extracted. Figure 5.11 shows the power of stability in discriminating such patterns. The concept with intent $\langle(CL)\rangle$ *and* $\langle(\top_{hospital}) \times 2\rangle$ is the most frequent. It represents patients with at least a stay in a private organization, and at least 2 stays in hospital. Similar concepts have a relatively close support, and differ only in the total number of stays. The concept with 5 stays has the highest stability. This probably matches the basic treatment scheme of lung cancer. Our interpretation relies on the power of stability to point out noisy concepts. Actually, only a few patients in concept $\langle(CL)\rangle$ *and* $\langle(\top_{hospital}) \times 2\rangle$ have only 2 stays.

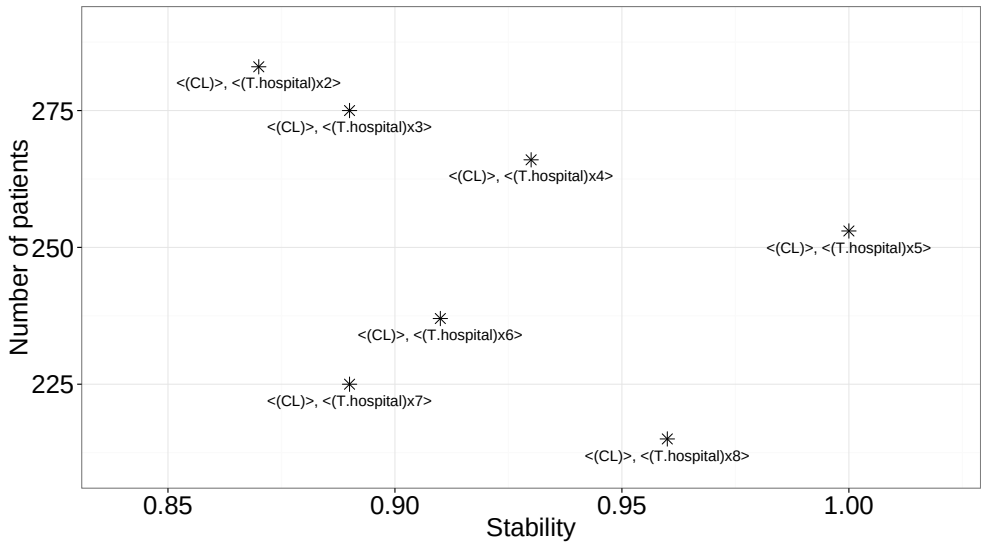


Figure 5.11: Discriminating power of stability: scatter plot of support and stability of concepts (represented by their intent)

5.6 Clustering Healthcare Trajectories by using sim_{ACS}

5.6.1 Clustering Healthcare Trajectories

In this chapter we cluster patient healthcare trajectories based on our similarity measure sim_{ACS} to compute degree of the similarity between them. In this experiment, every hospitalization can be described by itemset where the items represent the healthcare institution where it takes place, its main cause (diagnosis) and a set of medical and surgical procedures underwent by the patient. For example $\{Moselle, Metz\ regional\ hospital,\ lung\ cancer,\ chest\ \ radiography\}$ represents a stay in the regional hospital of the city of Metz, in the administative area of Moselle[10], for a lung cancer, where the patient underwent a chest radiography. A patient trajectory is modeled as a sequence of itemsets, each itemset representing one hospitalization. Computing similarity between patient care trajectories will open the way to patient clustering.

Our dataset contains the same 828 patients which are suffering from lung cancer and live in the Lorraine region, of Eastern France. Figure 5.12 shows the distribution of the length of sequences of care in this dataset, the median length being 54 (median size is equal to 11 stays).

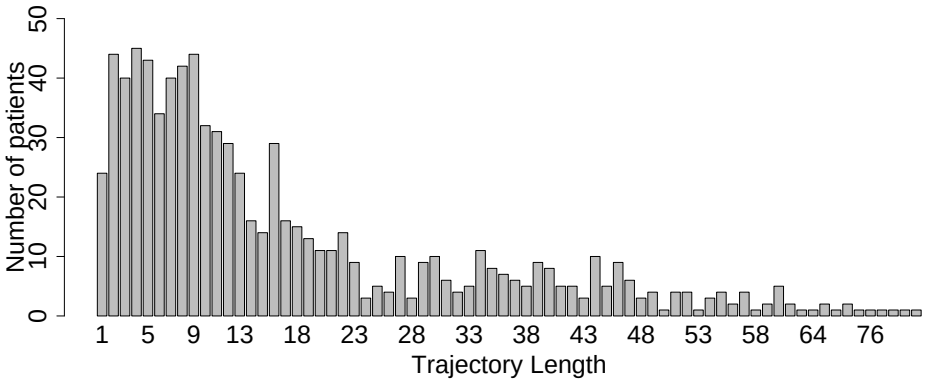


Figure 5.12: Distribution of the length of patient trajectories

Table 5.6 shows an example of healthcare trajectories for four patients. For example, P_1 has two hospitalizations. He was admitted in the University Hospital of Nancy (encoded as CHU_{nancy}), in Meurthe-et-Moselle (department number 54) for a lung cancer ($C34$), and underwent a chest radiography ($ZBQK$). Then, he was hospitalized in a private clinic in the city of Metz (CL_{metz}), Moselle (department number 57), for a chemotherapy session ($Z51$) where he also had a chest radiography.

Our similarity measure sim_{ACS} was used to build a similarity matrix between patient trajectories. A hierarchical clustering procedure was then applied using the *hclust* method in R software R Core Team [2012]. The number of clusters was set to 4 based on a priori knowledge from our experts. To assess the quality of our similarity measure, we describe each cluster with *"representative"* trajectories. To do so, we first extract frequent closed sequential patterns from our dataset by applying CloSpan Yan et al. [2003] with a minimal support of 10%. Then, the support of the obtained patterns is computed in each of the 4 different clusters. Patterns having the highest variation of support between clusters were detected using a chi-squared measure (χ^2).

[10]Moselle is one of the 101 departments of France

Patients	*Trajectories*
P_1	$\langle\{54, CHU_{nancy}, C34, ZBQK\}\{57, CL_{metz}, Z51, ZBQK\}\rangle$
P_2	$\langle\{54, CHU_{nancy}, I70, ZBQK, GFFA\}\{67, CL_{strasbourg}, Z51, GFFA\}\rangle$
P_3	$\langle\{75, CH_{paris}, C34, ZBQK\}\{57, CL_{metz}, Z51, GFFA, GLLD\}\rangle$
P_4	$\langle\{54, CHU_{nancy}, K24, ZBQK, GFFA\}\{57, CL_{metz}, C34, GFFA\}\rangle$

Table 5.6: Healthcare trajectories of 2 patients

Patterns with a high χ^2 and a high support in a given cluster are considered as distinguishing features of that cluster. After discussing these results with our medical expert, two criteria appeared to be related with the results of the clustering process, the *place of hospitalization* and the *length of the care trajectories*. We describe in the following the different clusters built with our similarity measure and its associated medical explanations.

1. The pattern $\langle\{54, C34, GFFA\}\rangle$ has a high χ^2 and a high support in Cluster 1. Patients in that cluster underwent a pneumonectomy ($GFFA$) in a hospital from Meurthe-et-Moselle (department number 54). Patients usually have a short trajectory (median is 6 stays).

2. The pattern $\langle\{57, C34, GFFA\}\rangle$ is highly frequent in the second cluster (support is around 80%) but not in the others. It contains patients having underwent a pneumonectomy ($GFFA$) in a hospital from Moselle (department number 57). This cluster is characterized by longer patterns with repeated stays in the departement of Moselle, such as $\langle\{57\}\{57\}\{57\}\{57\}\{57\}\{57\}\rangle$. Patients in that cluster have a median trajectory length of 13.

3. The pattern $\langle\{54, Z51\}\{54, Z51\}\{54, Z51\}\{54, Z51\}\{54, Z51\}\{54, Z51\}\rangle$ is over represented in the third cluster (support is approximately 95%) than in any other cluster. It represents patients who have repeated chemotherapy sessions in Meurthe-et-Moselle department. The median trajectory length in that cluster is 37.

4. This cluster is similar to the third cluster (chemotherapy sessions) but with stays occurring in various places, especially in the border region of Alsace.

The clustering is based on a combination of different trajectory lengths and precise diagnoses or procedures such as pneumonectomy or chemotherapies. Our similarity measure is only based on the number of common subsequences and we were able to build clusters that were close to the knowledge that doctors and experts have on patients trajectories in the Lorraine Region. Furthermore, for our experts, these results are very encouraging as they correspond to the two main modalities in care for lung cancer: (i) surgery only or (ii) chemothery with or without surgery. They also highlight some important geographical characteristics in care trajectories and suggest that variability in the organization of cancer care is related to local factors.

5.6.2 A Comparison Between sim_{ACS} and sim_{LCS}

We compare sim_{ACS} with the two sim_{LCS} similarity measures (i.e., using either sequence length or size). For a given sequence S, we ranked the 828 other sequences according to their similarity with S. Agreement between sim_{ACS} and sim_{LCS} rankings was then analyzed using Spearman's rank correlation coefficient. Figure 5.13 shows the link between a sequence length and its related Spearman coefficient Myers and Well [2003]. The agreement is high (Spearman $\rho > 0.8$) for short

and long sequences. For medium length sequences, sim_{ACS} and sim_{LCS} have a very different behavior. In Figure 5.13, sim_{LCS} has been computed using sequence length, but the same comments apply to the measure based on sequence size. A deeper comparison was performed by selecting a random sample of patient trajectories. For each sequence, rank correlation between similarity measures was analyzed with a scatter plot. An example of the similarity rankings is given in Table 5.7 for patient P_{656}. The scatter plots uses the sim_{ACS} ranking as an X-axis and the $sim_{LCS_{length}}$ ranking values for the Y-axis. Two different distributions of similarity values appear as shown in Figures 5.14a and 5.14b.

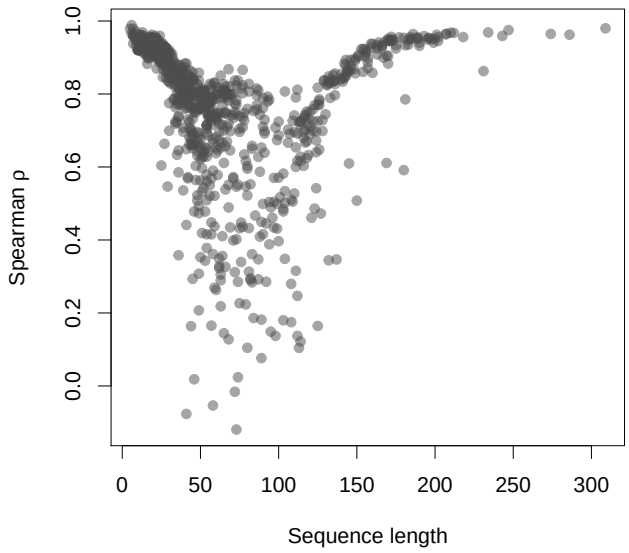
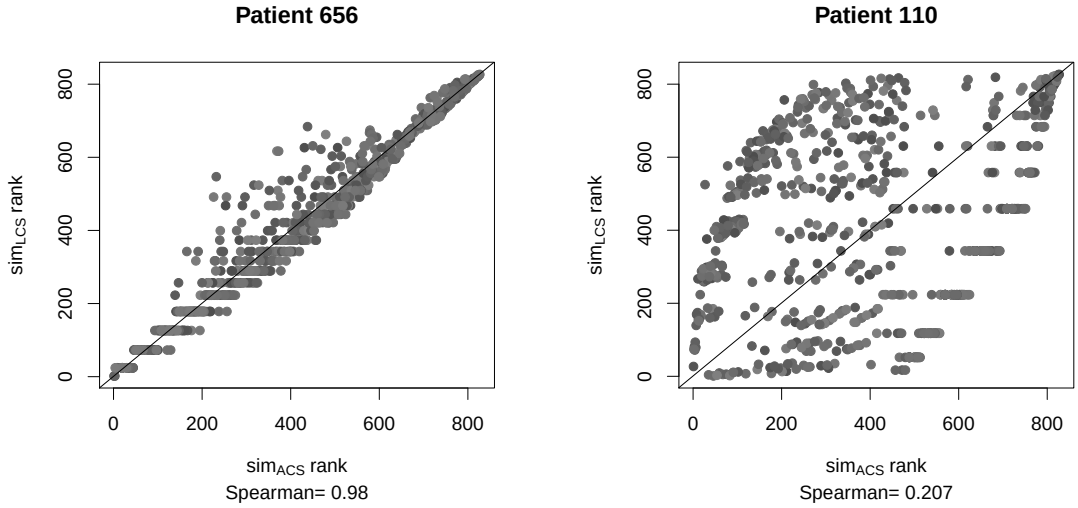


Figure 5.13: Agreement between sim_{ACS} versus sim_{LCS} according to sequence length.

Patient	sim_{ACS} ranking	$sim_{LCS_{length}}$ ranking
P_{40}	291	468
P_{502}	126	126
P_{209}	484	491
⋮	⋮	⋮
P_{827}	129	126

Table 5.7: Example of rank correlation for patient P_{656}

Strong correlation between sim_{ACS} and sim_{LCS} results. This case corresponds in our data set to the shortest and longest sequences, which have similar results for sim_{ACS} and sim_{LCS} (the optimal scatter plot in this case is a perfect diagonal). Their similarity with sequences of a much different length is very small and close to zero. This is the case, for example on Figure 5.14a, with patient 656 who went through 99 hospitalizations (one of the longest trajectories). Actually, whether similarity is measured with sim_{ACS} or sim_{LCS}, it decreases monotically when the size



(a) Strong correlation (b) Weak correlation

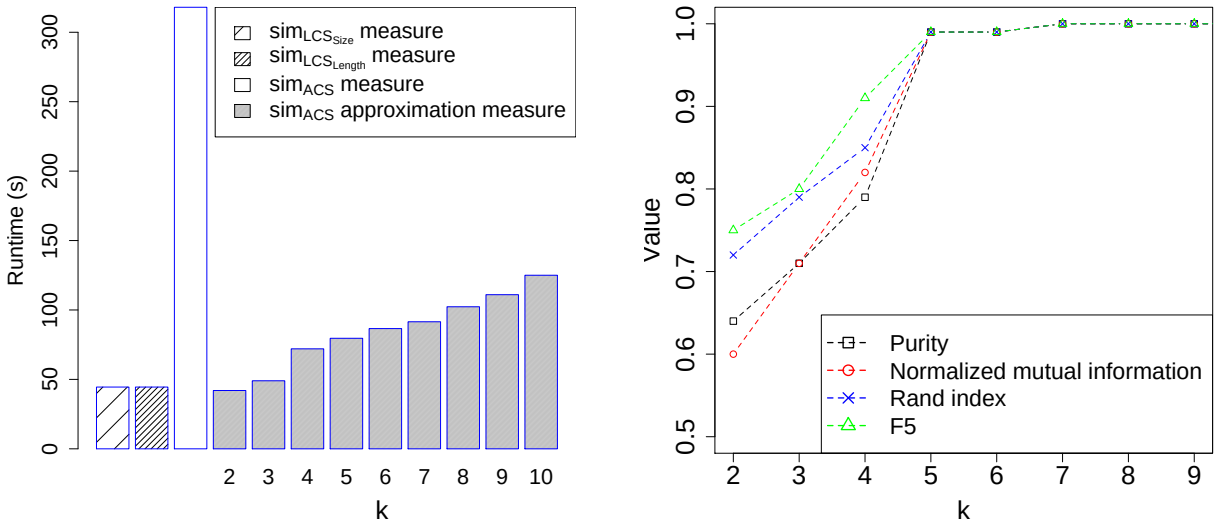
Figure 5.14: sim_{ACS} versus sim_{LCS}

or the length difference between the two sequences increases. It can also be noticed that there is a high number of *ties* for sim_{LCS} (i.e., a repetition of the same similarity value for different sequences) as it can be seen on the plot in Figure 5.14a with different strata (i.e., bands).

Weak correlation between sim_{ACS} and sim_{LCS} results. This case corresponds to medium length trajectories, with two possible situations, as shown in Figure 5.14b. First situation: sim_{ACS} rankings are smaller than sim_{LCS} rankings (upper left part of the scatter plot). This happens when two patients have a long common subsequence but still have differences in some itemsets (i.e., hospitalizations). As sim_{ACS} is more sensitive to items variability it tends to output smaller similarity values than sim_{LCS}. Second situation: sim_{ACS} rankings are higher than sim_{LCS} rankings (lower right part of the scatter plot). This can be observed when two patients have similar hospitalizations but differ in their order (i.e., permutations). On the one hand, these differences tend to break long common subsequences and produce a small sim_{LCS} value. On the other hand, there is still a high number of common subsequences which is reflected through the higher sim_{ACS} similarity.

5.6.3 Linial-Nisan Approximation

The runtime to build the similarity matrix for 828 patient trajectories without approximation is about 5 minutes. We applied the Linial-Nisan approximation on the PMSI dataset and built the similarity matrix with several values of $k \in [2, 10]$. Figure 5.15a shows the computation time to build the similarity matrix with varying values for k. To assess the quality of the approximations, we compare the clusters obtained using the Linial-Nisan approximation with the clusters obtained using sim_{ACS}. We use several criteria of clustering quality (i.e, Purity, Normalized mutual information, Rand index and F measure) proposed in Manning et al. [2008] to evaluate the quality of the clusters obtained using the Linial-Nisan approximation. Figure 5.15b shows the results with several values of k. The higher the value of a cluster quality is, the more homogeneous the cluster is (i.e., it contains similar objects to the previously computed cluster



(a) Runtime for building the similarity matrix for 828 patient trajectories

(b) Clusters matching values obtained with the Linial-Nissan approximation

Figure 5.15: Linial-Nisan cluster matching and runtime

with sim_{ACS}). This highlights the fact that approximating our similarity measure still yields good and competitive conclusions, with fast computation times (less than 120 seconds).

5.7 Conclusion

In this chapter we design a system for mining patients healthcare trajectories. This system is based on several methods proposed in this thesis. It helps in the detection, classification and clustering of patient healthcare trajectories with an application to the oncology domain. This system gives potential interesting results for healthcare specialists. These results represented in several forms such as sequential patterns which can be seen as a condensed representation of care trajectories or in a form of groups of patient sharing some attributes during their trajectories. After discussing these results with our medical expert, we found that they are very encouraging as they correspond to several modalities in care for cancer diseases and they can also characterize group of patients by several trajectories.

As we can see that mining patients healthcare trajectories generates a large number of results. Knowledge extraction from these results is a difficult task for the healthcare managers and decision makers. In order to make our results more understandable, we are planning to focus more deeply on complementing this mining system with a graphical interface to visualize and query the results. As a first step, we built a web page to visualize different clusters of patient built with our similarity measure, which also allow user interaction. For example, figure 5.16 shows the four clusters of patient built with our similarity measure. In this figure, each small circle represents a patient trajectory. The four big circles represent the four clusters. Each cluster is further divided into sub-clusters depending on the type of the first principal diagnosis which appears in patient trajectory. The left big circle represents one group of patients where

the major principle diagnosis of them are treating the *Neoplasms (C)*, *Blood problem (D)* and for doing *Chemotherapy (Z)*. Actually, this kind of visualization helps the healthcare managers to extract knowledge form each clusters and give a general representation for each group of patient. For further details, visit `http://webloria.loria.fr/~eegho/acs/pmsi.html#`. As a future work, we are planing to visualize other results extracted by another method like the frequent trajectories extracted by *MMISP* and *Sequential Pattern Structure*.

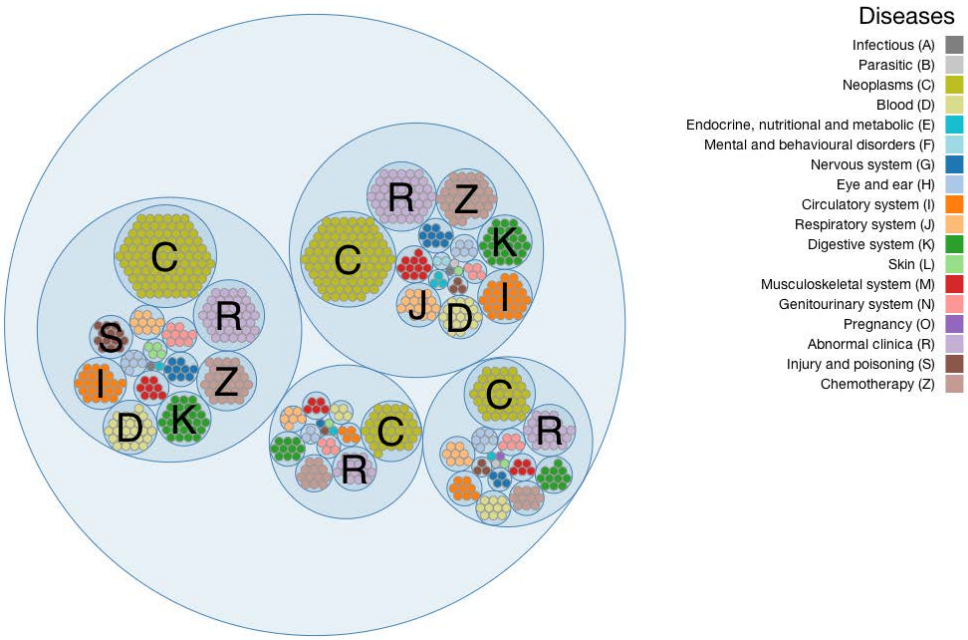


Figure 5.16: Visualization of the patient clusters

Chapter 6

Conclusion and Perspectives

Contents

6.1 Summary of Contributions 121
6.1.1 Extraction of Multidimensional and Heterogeneous Sequential Patterns 122
6.1.2 Sequential Pattern Structure 122
6.1.3 Classification of Multidimensional and Heterogeneous Sequential Patterns: 123
6.1.4 Clustering Of Sequences: 123
6.2 Perspective 124
6.3 List of publications 125

This thesis sets out several topics related to sequential database mining. Mining task is not a monolithic and univocal process in which it would apply a general principle to all the sequential data. Depending on the types of information to be found, mining task can be derived in several forms such as extraction of patterns, classification and segmentation. Chapter 1 and Chapter 4 present several methods to mine classical sequential database. In real-world applications, e.g. in medical or biology, one rarely comes across a classical sequential data, where heterogeneous and multidimensional sequential database are more typical. In this kind of sequential data, each event combines several dimensions described over different levels of granularity. This kind of data is called a heterogeneous multidimensional sequential database, for instance the patients trajectories in a healthcare system are a good example of such kind of sequential data.

This thesis contributes to designing a system for mining heterogeneous multidimensional sequential data. It helps in the detection, classification and clustering of patient healthcare trajectories with an application to the oncology domain. It analyses data from a medico-administrative database which consists of information about the hospitalizations of patients in Lorraine, France. The contributions allow the identification of characteristics of the episodes of care. The goal of this chapter is to give an overview of all the proposed methods, conclude this thesis and state the main open questions following from our research.

6.1 Summary of Contributions

In this thesis, we are interested in the sequential pattern mining on healthcare patient trajectories. In order to achieve this goal, four different research directions were studied:

- Extraction of sequential patterns representing the healthcare patients trajectories

- Using sequential pattern structure to mine patients trajectories.
- Classification of healthcare patients trajectories.
- Clustering healthcare patients trajectories.

6.1.1 Extraction of Multidimensional and Heterogeneous Sequential Patterns

The main contribution of this part of the thesis is to generalize the concept of multidimensional sequence which is defined by Pinto et al. [2001] by considering a more complex and heterogeneous structure. These kind of data are called heterogeneous multidimensional sequential data where the sequence is often represented as an ordered list of vectors with elements having different types (i.e., *item* and *itemset*) where each element can be represented at different levels of granularity, by using a taxonomy as background knowledge. Such multidimensional and heterogeneous sequential data has to be mined by adapted and suitable methods. We propose a new method *MMISP (Mining Multidimensional Itemsets Sequential Patterns)* to extract sequential patterns from this heterogeneous multidimensional sequential data by taking into account the background knowledge which is in the form of taxonomies. As often with enumeration algorithms, extracting all possible sequential patterns from a multidimensional sequential database results in a huge amount of patterns which is difficult to be analyzed. To overcome this problem, *MMISP* mines only the most specific sequential patterns. In Chapter 5, we apply *MMISP* method on a set of patients to extract some frequent healthcare trajectories. The patterns extracted by *MMSIP* help healthcare managers and deciders in planning and organizing healthcare resources at a regional level. Besides, we present how sequential patterns can be seen as a condensed representation of care trajectories. We compare our method, *MMISP*, with two existing methods *CloSpan* and M^3SP. We see that *MMISP* extracts more condensed trajectories as compare to the trajectories extracted by *CloSpan* and M^3SP. In addition, some frequent trajectories can be extracted by *MMISP* but they can not be extracted by M^3SP. Other experiments have been conducted on synthetic datasets to study the scalability of the *MMISP* approach. We show that *MMISP* is efficient in terms of runtime for a large panel of sequences with varying different parameters.

6.1.2 Sequential Pattern Structure

In this part, we propose a novel way of dealing with heterogeneous multidimensional sequences by mapping them to pattern structures Ganter and Kuznetsov [2001b]. The problem with *MMISP* (and generally with all pattern enumeration algorithms) is that it generates a large number of sequential patterns while few of them are truly relevant. With the help of sequential pattern structure, we can solve this problem by developing a framework for enumerating only patterns of required types based solely on concept lattices and its associated measures. In our approach, we consider only the order of consequent elements in the sequences as patterns. Such kind of extracted sequences are called consecutive sequences. The meet-semilattice operation between two sets of sequences is defined as the set of all maximal common consecutive subsequences between them. To find all common consecutive subsequences of two sequences, we first align the two sequences in all possible way, then we keep only the maximal ones. In this proposition, we adapted *AddIntent* algorithm Van Der Merwe et al. [2004] to process pattern structures instead of standard FCA contexts by substituting every set intersection operation on attributes with our sequential meet-semilattice operation. Pattern structures in term give rise to many computational challenges. To solve this problem, we introduce and discuss the notion of *projections* for sequential pattern structures. These mathematical objects significantly help us to decrease the

number of patterns, while preserving the most interesting ones for an expert. We also show that projections are easily built to answer questions that an expert may have and also to provide an efficient tool for the analysis of heterogeneous multidimensional sequential data. Chapter 5 discusses the application of sequential pattern structure to mine patients healthcare trajectory. Using the *sequential pattern structure* as a mining method and *projection* as a filter help us in extracting and selecting some interesting frequent healthcare trajectories which answer the questions of healthcare managers and decision makers.

6.1.3 Classification of Multidimensional and Heterogeneous Sequential Patterns:

In this part we propose an original combination of heterogeneous multidimensional sequential pattern mining and FCA. We propose use FCA as lattice-based classification method to classify the sequential patterns mined with *MMISP*. We show some interesting properties of concept lattices and stability index to classify them in form of concept lattice, then to select interesting sequential patterns with the help of support and stability measures. In Chapter 5, we conduct an experimentation over healthcare dataset to classify the patient healthcare trajectories. This experiment provides two advantages. First, it helps in grouping patient trajectories sharing some attributes and producing their graphical visualizations. The second benefit of using FCA is that it is able to characterize group of patients by several trajectories.

6.1.4 Clustering Of Sequences:

Chapter 4 studies the problem of counting all common subsequences between two sequences of itemsets. Theoretical results and an efficient dynamic programming algorithm are presented to count the number of common subsequences between two sequences. This solution allows us to define in a simple and intuitive manner a similarity measure, denoted as sim_{ACS}, between two sequences S and T. We notice that sim_{ACS} relies heavily on the inclusion-exclusion principle. The computation drawback is the fact that the inclusion-exclusion formula has an exponential number of terms which can become a problem with very long sequences. This prompted our interest in approximating our similarity measure through the approximation of the inclusion-exclusion formula used in both ACS and ADS computations. To solve this problem and to cope with large data sets containing long input sequences, an *approximation technique* is proposed to compute the similarity efficiently. The approach relies on approximating the size of a union of a family of sets in terms of the intersections of all subfamilies (i.e., *inclusion-exclusion principle*) based on the direct application of a result from Linial and Nisan [1990].

We compare conceptually between *longest common subsequence* and *all common subsequences* to compute the similarity between two sequences. We noticed that *longest common subsequence* can lead to *ambiguous* results and *all common subsequences* is very *sensitive* to sequence modification when adding new item or new itemset as compared to *longest common subsequence*.

With the help of experiments, we study the qualitative results of our similarity measure sim_{ACS} by conducting two real experiments on classification of patient healthcare trajectories and on the online dataset about handwritten Assamese characters. The first experiment shows that with our similarity, we were able to build clusters that are close to the knowledge that doctors and experts have on patients trajectories. Additionally, we show that the results observed are very encouraging as they correspond to the two main modalities in care for lung cancer: (i) surgery only or (ii) chemothery with or without surgery. They also highlight some important geographical characteristics in care trajectories and suggest that variability in the organization

of cancer care is related to local factors.

Longest common subsequence and *all common subsequences* are compared to cluster 8235 symbols collected from 45 users in online handwritten Assamese characters dataset. The symbols were considered as a sequence of itemsets, then we apply clustering by using two matrices of similarity the first one was built with sim_{ACS}, while the another one is built by using sim_{LCS}. We finally observed that clustering with sim_{ACS} returns good results with a clear partition of the similar symbols and more homogeneous clusters. For the medical and handwritten datasets, we compare the clusters obtained using the Linial-Nisan approximation with the ones obtained using sim_{ACS}. We conclude that the clusters built using the Linial-Nisan approximation are identical to the clusters built using sim_{ACS}. The visualization of the results obtained by our similarity measure are available at `http://www.loria.fr/~eegho/acs/`.

6.2 Perspective

The current thesis has passed way for a variety of prospectives. We describe here some perspectives and we will discuss the short-term outlook to improve each method in our system to mine patient healthcare trajectory. This improvement can be either in the efficiency or in the quality of the extracted knowledge.

In Chapters 2, we propose a method *MMISP* to mine heterogeneous multidimensional sequential data and extract sequential patterns from the most specific elementary vectors.This method can be further extended by extracting sequences from elementary vectors having other features. For example, we can extract sequential patterns from the elementary vectors which are closed or generator. Such an extraction will help us to extract not just the most specific sequential patterns.

In *MMISP*, we define hierarchies as taxonomies. In practice, hierarchies are generally directed graph with no directed cycles (DAG). The use of DAG in our proposal require some adjustments. In a DAG, there may be two different paths from the root to a node of the graph. It is important to note that each path is associated with a specific semantic. Integration of hierarchies as DAGs must take into account the semantics of each path. Thus, a preferred path relative to another can help to prune the search space faster and easy adaption of the algorithm *MMISP*.

As presented before, *MMISP* generates a large number of sequential patterns, thus, we are also planning to use statistical significance tests such as statistical hypothesis testing to evaluate the sequential patterns extracted and choose the k most significant ones.

It has also been observed that the performance of *MMISP* was challenged when the number of elements or the number of levels in the taxonomy for each element were too high. It would be very interesting to work on reducing the number of elements or the number of levels in the taxonomy before doing the extraction of sequential patterns. It can for example be based on detecting dependencies between elements to remove elements which depend on other. We can also prune the taxonomy by removing some levels in the taxonomy and going directly to the most general node. We can also rely on the extraction of sequential patterns by approximation. If we accept not having the exact solution, approximations are an excellent solution. There is some assseactly existing work about the extraction of sequences approximation based on sampling data sets Luo and Chung [2008]. It would be interesting to adapt these approaches in a heterogeneous multidimensional context.

In Chapter 3, we present a *sequential pattern structure* and *projection* to mine heterogeneous multidimensional sequential data. For future work, we are planning to more deeply investigate projections, their potentialities w.r.t. the types of patterns. We can apply *sequential pattern*

structure not just to mine a heterogeneous multidimensional sequence but to mine more complex structure like sequences of graphs. Another part of this work applies FCA to classify and qualify the sequential patterns extracted by *MMISP*. In this part, connection between FCA and the sequential mining problem could be explored in a more integrative approach, especially by studying closure operators on sequences.

In Chapter 4, we present a similarity measure between two sequences. We apply the similarity on two different domains (healthcare trajectories and handwritten Assamese characters). As a future work, we plan to apply the measure on various sequence data sets (more precisely trajectory mining, molecular bioinformatics and text classification). We also intend to compare our measure with kernel methods for sequence classification. We are interested also in counting all the common subsequences under some constraint like counting all the common subsequences with a length that is smaller than k or all the common subsequences with a minimum or maximum gap. We are interested in making our measure more general by proposing a formal to count all the common subsequence between two sequences of graphs.

In Chapter 5, we present an application of our sequential mining system to mine patients healthcare trajectories. We present that extraction the knowledge from the results of this system is a difficult task for the healthcare managers and decision makers. In this thesis, we visualized different clusters of patient built with our similarity measure. As a future work, we are planing to visualize other results extracted by another method like the frequent trajectories extracted by *MMISP* and *Sequential Pattern Structure*.

6.3 List of publications

International peer-reviewed journals

Elias Egho, Chedy Raïssi, Toon Calders, Nicolas Jay, Amedeo Napoli: On Measuring Similarity for Sequences of Itemsets, Data Mining and Knowledge Discovery (DAMI).

Elias Egho, Nicolas Jay, Chedy Raïssi, Dino Ienco, Pascal Poncelet, Maguelonne Teisseire, Amedeo Napoli: A contribution to the discovery of multidimensional patterns in healthcare trajectories. J. Intell. Inf. Syst. 42(2): 283-305 (2014).

Book chapters

Elias Egho, Chedy Raïssi, Dino Ienco, Nicolas Jay, Amedeo Napoli, Pascal Poncelet, Catherine Quantin, Maguelonne Teisseire: Healthcare Trajectory Mining by Combining Multidimensional Component and Itemsets. New Frontiers in Mining Complex Patterns: 109-123.

International peer-reviewed conferences

Elias Egho, Chedy Raïssi, Nicolas Jay, Amedeo Napoli: Mining Heterogeneous Multidimensional Sequential Patterns. ECAI 2014.

Elias Egho, Nicolas Jay, Chedy Raïssi, Gilles Nuemi, Catherine Quantin, Amedeo Napoli: An Approach for Mining Care Trajectories for Chronic Diseases. AIME 2013: 258-267.

Aleksey Buzmakov, **Elias Egho**, Nicolas Jay, Sergei O. Kuznetsov, Amedeo Napoli, Chedy

Raïssi: On Projections of Sequential Pattern Structures (with an Application on Care Trajectories). CLA 2013: 199-208.

Elias Egho, Nicolas Jay, Chedy Raïssi, Amedeo Napoli: A FCA-based Analysis of Sequential Care Trajectories. CLA 2011: 363-376.

National peer-reviewed conferences

Elias Egho, Chedy Raïssi, Toon Calders, Thomas Bourquard, Nicolas Jay, Amedeo Napoli: Vers une mesure de similarité pour les séquences complexes. Extraction et gestion des connaissances (EGC 2013): 335-340.

Workshops

Aleksey Buzmakov, **Elias Egho**, Nicolas Jay, Sergei O. Kuznetsov, Amedeo Napoli and Chedy Raïssi. FCA and pattern structures for mining care trajectories. Fornal Concept Analysis for Artificial Intelligence (FCA4AI 2013) workshop at International Conference on Artificial Intelligence (IJCAI 2013).

Aleksey Buzmakov, **Elias Egho**, Nicolas Jay, Sergei O. Kuznetsov, Amedeo Napoli and Chedy Raïssi: The representation of sequential patterns and their projections within Formal Concept Analysis. Languages for Data Mining and Machine Learning (LML 2013) workshop at European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. (ECML/PKDD 2013).

Elias Egho, Chedy Raïssi, Dino Ienco, Nicolas Jay, Amedeo Napoli, Pascal Poncelet, Catherine Quantin, Maguelonne Teisseire: Healthcare Trajectory Mining by Combining Multidimensional Component and Itemsets. New Frontiers in Mining Complex Patterns workshop at European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. (ECML/PKDD 2012).

Bibliography

John Aach and George M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

C. C. Aggarwal and P. S. Yu. Mining associations with the collective strength approach. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):863–873, November 2001. ISSN 1041-4347. doi: 10.1109/69.971183. URL `http://dx.doi.org/10.1109/69.971183`.

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8. URL `http://dl.acm.org/citation.cfm?id=645920.672836`.

Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Advances in knowledge discovery and data mining. chapter Fast Discovery of Association Rules, pages 307–328. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996. ISBN 0-262-56097-6. URL `http://dl.acm.org/citation.cfm?id=257938.257975`.

Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *KDD*, pages 429–435, 2002.

José L. Balcázar and Gemma Casas-Garriga. On horn axiomatizations for sequential data. In *ICDT*, pages 215–229, 2005.

José L. Balcázar and Gemma C. Garriga. Horn axiomatizations for sequential data. *Theor. Comput. Sci.*, 371(3):247–264, 2007.

M. Barbut and B. Monjardet. *Ordre et classification: algèbre et combinatoire*. Collection Hachette université: Méthodes mathématiques des sciences de l'homme. Hachette, 1970. URL `http://books.google.fr/books?id=n3BpSgAACAAJ`.

Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Computational Logic*, pages 972–986, 2000.

Iyad Batal, Lucia Sacchi, Riccardo Bellazzi, and Milos Hauskrecht. A temporal abstraction framework for classifying clinical temporal data. *AMIA Annu Symp Proc*, 2009:29–33, 2009.

Roberto J. Bayardo, Jr. Efficiently mining long patterns from databases. *SIGMOD Rec.*, 27(2): 85–93, June 1998. ISSN 0163-5808. doi: 10.1145/276305.276313. URL `http://doi.acm.org/10.1145/276305.276313`.

Roberto J. Bayardo, Jr., Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. *Data Min. Knowl. Discov.*, 4(2-3):217–240, July 2000. ISSN 1384-5810. doi: 10.1023/A:1009895914772. URL `http://dx.doi.org/10.1023/A:1009895914772`.

Richard Bellman, John H. Holland, and Robert Kalaba. On an application of dynamic programming to the synthesis of logical systems. *J. ACM*, 6(4):486–493, 1959.

Lasse Bergroth, Harri Hakonen, and Timo Raita. A survey of longest common subsequence algorithms. In *String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. Seventh International Symposium on*, pages 39–48. IEEE, 2000.

Donald J. Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD Workshop*, pages 359–370, 1994.

Kevin S. Beyer and Raghu Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 359–370. ACM Press, 1999. ISBN 1-58113-084-8.

Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, pages 39–48, 2003.

G. Birkhoff. *Lattice Theory*. Colloquium publications. American Mathematical Soc., 1964. ISBN 9780821889534. URL `http://books.google.fr/books?id=o4bu3ex9BdkC`.

Jean-Paul Bordat. Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et Sciences humaines*, 96:31–47, 1986.

Christian Borgelt. Simple algorithms for frequent item set mining. In *Advances in Machine Learning II*, pages 351–369. 2010.

Christian Borgelt and Xiaomeng Wang. Sam: A split and merge algorithm for fuzzy frequent item set mining. In *IFSA/EUSFLAT Conf.*, pages 968–973, 2009.

Christian Borgelt, Xiaoyuan Yang, Rubén Nogales-Cadenas, Pedro Carmona-Saez, and Alberto D. Pascual-Montano. Finding closed frequent item sets by intersecting transactions. In *EDBT*, pages 367–376, 2011.

O. Burdakov, A. Grimvall, M. Hussian, and O. Sysoev. Hasse diagrams and the generalized pav algorithm for monotonic regression in several explanatory variables. Technical report, 2005.

Doug Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *In ICDE*, pages 443–452, 2001.

EG Caiani, A Porta, G Baselli, M Turiel, S Muzzupappa, F Pieruzzi, C Crema, A Malliani, and S Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. In *Computers in Cardiology 1998*, pages 73–76. IEEE, 1998.

Gemma Casas-Garriga. Summarizing sequential data with closed partial orders. In *SDM*, 2005.

Michel Chein. Algorithme de recherche des sous-matrices premiéres dúne matrice. *Bull. Math. Soc. Sc. Math. de Roumanie*, 1(13):21–25, 1969.

C. Chothia and M. Gerstein. Protein evolution. how far can sequences diverge? *Nature*, 6617 (385):579–581, 1997.

Victor Codocedo, Ioanna Lykourentzou, and Amedeo Napoli. A semantic approach to concept lattice-based information retrieval. *Annals of Mathematics and Artificial Intelligence*, pages 1–27, 2014. ISSN 1012-2443. doi: 10.1007/s10472-014-9403-0. URL `http://dx.doi.org/10.1007/s10472-014-9403-0`.

Shengnan Cong, Jiawei Han, and David Padua. Parallel mining of closed sequential patterns. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 562–567, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081937. URL `http://doi.acm.org/10.1145/1081870.1081937`.

Ayhan Demiriz. webspade: A parallel sequence mining algorithm to analyze web log data. In *ICDM*, pages 755–758, 2002.

Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. Springer, 2009.

William DuMouchel and Daryl Pregibon. Empirical bayes screening for multi-item associations. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 67–76, New York, NY, USA, 2001. ACM. ISBN 1-58113-391-X. doi: 10.1145/502512.502526. URL `http://doi.acm.org/10.1145/502512.502526`.

Margaret H Dunham. *Data mining: Introductory and advanced topics*. Pearson Education India, 2006.

Cees Elzinga, Sven Rahmann, and Hui Wang. Algorithms for subsequence combinatorics. *Theor. Comput. Sci.*, 409(3):394–404, 2008.

Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, SIGMOD '94, pages 419–429, New York, NY, USA, 1994. ACM.

Usama Fayyad, Gregory Piatetsky-shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 1996a.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996b.

Robert B Fetter, Youngsoo Shin, Jean L Freeman, Richard F Averill, and John D Thompson. Case mix definition by diagnosis-related groups. *Medical care*, pages i–53, 1980.

Bernhard Ganter. Two basic algorithms in concept analysis. In *ICFCA*, pages 312–340, 2010.

Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In *ICCS*, pages 129–142, 2001a.

Bernhard Ganter and Sergei O Kuznetsov. Pattern structures and their projections. In *Conceptual Structures: Broadening the Base*, pages 129–142. Springer, 2001b.

Bernhard Ganter and Klaus Reuter. Finding all closed sets: A general approach. *Order*, 8(3): 283–290, 1991.

Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg, 1999.

Minos N Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Spirit: Sequential pattern mining with regular expression constraints. In *VLDB*, volume 99, pages 7–10, 1999.

DM Gavrila and LS Davis. Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In *International workshop on automatic face-and gesture-recognition*, pages 272–277. Citeseer, 1995.

K Gollmer and C Posten. Detection of distorted pattern using dynamic time warping algorithm and application for supervision of bioprocesses. *On-line fault detection and supervision in chemical process industries*, 1995.

Antonio Gomariz, Manuel Campos, Roque Marín, and Bart Goethals. Clasp: An efficient algorithm for mining frequent closed sequences. In *PAKDD (1)*, pages 50–61, 2013.

Gösta Grahne and Jianfei Zhu. Fast algorithms for frequent itemset mining using fp-trees. *IEEE Trans. Knowl. Data Eng.*, 17(10):1347–1362, 2005.

Nicola Guarino. Formal ontology, conceptual analysis and knowledge representation. *International journal of human-computer studies*, 43(5):625–640, 1995.

Valerie Guralnik and George Karypis. Parallel tree-projection-based sequence mining algorithms. *Parallel Computing*, 30(4):443–472, 2004.

Jiawei Han and Yongjian Fu. Mining multiple-level association rules in large databases. *Knowledge and Data Engineering, IEEE Transactions on*, 11(5):798 –805, sep/oct 1999. ISSN 1041-4347. doi: 10.1109/69.806937.

Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *KDD*, pages 355–359, 2000a.

Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000b. ISSN 0163-5808. doi: 10.1145/335191.335372. URL `http://doi.acm.org/10.1145/335191.335372`.

Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

Tutut Herawan and Mustafa Mat Deris. A soft set approach for association rules mining. *Knowledge-Based Systems*, 24(1):186–195, 2011.

Javier Herranz, Jordi Nin, and Marc Sole. Optimal symbol alignment distance: A new distance for sequences of symbols. *IEEE Transactions on Knowledge and Data Engineering*, 23:1541–1554, 2011.

D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, June 1975.

Daniel S Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, 24(4):664–675, 1977.

Jun Huan, Wei Wang, Jan Prins, and Jiong Yang. Spin: mining maximal frequent subgraphs from graph databases. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–586. ACM, 2004.

Zhengxing Huang, Xudong Lu, and Huilong Duan. On mining clinical pathway patterns from medical behaviors. *Artif Intell Med*, 56(1):35–50, Sep 2012. doi: 10.1016/j.artmed.2012.06.002. URL `http://dx.doi.org/10.1016/j.artmed.2012.06.002`.

Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

Nicolas Jay. *Découverte et représentation des trajectoires de soins par analyse formelle de concepts*. These, Université Henri Poincaré - Nancy I, October 2008. URL `http://tel.archives-ouvertes.fr/tel-00585411`.

Nicolas Jay, François Kohler, and Amedeo Napoli. Using formal concept analysis for mining and interpreting patient flows within a healthcare network. In *CLA*, pages 263–268, 2006.

Nicolas Jay, François Kohler, and Amedeo Napoli. Analysis of social communities with iceberg and stability-based concept lattices. In *ICFCA*, pages 258–272, 2008.

Nicolas Jay, Gilles Nuemi, Maryse Gadreau, and Catherine Quantin. A data mining approach for grouping and analyzing trajectories of care using claim data: the example of breast cancer. *BMC Med. Inf. & Decision Making*, 13:130, 2013.

Mohammed J. Zaki Karlton Sequeira. Admit: Anomaly-base data mining for intrusions. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2002.

Eamonn Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 406–417. VLDB Endowment, 2002.

Mingoo Kim, Hyunjung Shin, Tae Su Chung, Je-Gun Joung, and Ju Han Kim. Extracting regulatory modules from gene expression data by sequential pattern mining. *BMC Genomics*, 12 Suppl 3:S5, Nov 2011. URL `http://dx.doi.org/10.1186/1471-2164-12-S3-S5`.

Sotiris Kotsiantis and Dimitris Kanellopoulos. Association rules mining: A recent overview. *GESTS International Transactions on Computer Science and Engineering*, 32(1):71–82, 2006.

Zsolt Miklos Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22 (11):1266–1276, 2000.

Hye-Chung Kum, Jian Pei, Wei Wang, and Dean Duncan. Approxmap: Approximate mining of consensus sequential patterns. In *SDM*, 2003.

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.

Sergei Kuznetsov, Sergei Obiedkov, and Camille Roth. Reducing the representation complexity of lattice-based taxonomies. In *Conceptual Structures: Knowledge Architectures for Smart Applications*, pages 241–254. Springer, 2007.

Sergei O Kuznetsov. Stability as an estimate of the degree of substantiation of hypotheses derived on the basis of operational similarity. *Nauchn. Tekh. Inf., Ser*, 2(12):21–29, 1990.

Sergei O. Kuznetsov. Learning of simple conceptual graphs from positive and negative examples. In *PKDD*, pages 384–391, 1999.

Sergei O Kuznetsov. On stability of a formal concept. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):101–115, 2007.

Sergei O. Kuznetsov and Sergei Obiedkov. Comparing performance of algorithms for generating concept lattices. *JOURNAL OF EXPERIMENTAL AND THEORETICAL ARTIFICIAL INTELLIGENCE*, 14:189–216, 2002.

Philippe Lenca, Patrick Meyer, Benoit Vaillant, and Stephane 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.

Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for svm protein classification. *Pacific Symposium On Biocomputing*, 575(50):564–575, 2002.

V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

Yujian Li and Bi Liu. A normalized levenshtein distance metric. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1091–1095, 2007.

Nathan Linial and Noam Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.

Guimei Liu, Hongjun Lu, Wenwu Lou, and Jeffrey Xu Yu. On computing, storing and querying frequent patterns. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 607–612, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956827. URL `http://doi.acm.org/10.1145/956750.956827`.

Guimei Liu, Jinyan Li, and Limsoon Wong. A new concise representation of frequent itemsets using generators and a positive border. *Knowl. Inf. Syst.*, 17(1):35–56, October 2008. ISSN 0219-1377. doi: 10.1007/s10115-007-0111-5. URL `http://dx.doi.org/10.1007/s10115-007-0111-5`.

Congnan Luo and Soon M. Chung. A scalable algorithm for mining maximal frequent sequences using a sample. *Knowl. Inf. Syst.*, 15(2):149–179, May 2008. ISSN 0219-1377. doi: 10.1007/s10115-006-0056-0. URL `http://dx.doi.org/10.1007/s10115-006-0056-0`.

Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Florent Masseglia, Fabienne Cathala, and Pascal Poncelet. The psp approach for mining sequential patterns. In *PKDD*, pages 176–184, 1998.

Florent Masseglia, Pascal Poncelet, and Maguelonne Teisseire. Incremental mining of sequential patterns in large databases. *Data & Knowledge Engineering*, 46(1):97–121, 2003.

Harshitha Menon and Laxmikant V. Kalé. A distributed dynamic load balancer for iterative applications. In *SC*, page 15, 2013.

Roger Mitton. Ordering the suggestions of a spellchecker without using context. *Natural Language Engineering*, 15(2):173–192, 2009.

Tatsuo Motoyoshi, Hiroshi Kawakami, Takayuki Shiose, and Osamu Katai. Formal concept analysis of musicians' awareness for musical expression. 2009.

Mario E Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 108–115. IEEE, 1999.

Fariha Muzaffar, Bushra Mohsin, Farah Naz, and Lecturer Farooq Jawed. Dsp implementation of voice recognition using dynamic time warping algorithm. *IEEE Explore*, pages 1–7, 2005.

J. L. Myers and A. D. Well. *Research Design and Statistical Analysis*. Lawrence Erlbaum Associates, New Jersey, 2003.

Victoria Nebot and Rafael Berlanga. Finding association rules in semantic web data. *Knowledge-Based Systems*, 25(1):51–62, 2012.

E. M. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2):243–250, 1978.

Edward Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Trans. Knowl. Data Eng.*, 15(1):57–69, 2003.

Jose Oncina and Marc Sebban. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern Recogn.*, 39(9):1575–1587, September 2006.

Carlos Ordonez, Norberto Ezquerra, and Cesar A Santana. Constraining and summarizing association rules in medical data. *Knowledge and Information Systems*, 9(3):1–2, 2006.

Salvatore Orlando, Raffaele Perego, and Claudio Silvestri. A new algorithm for gap constrained sequence mining. In *SAC*, pages 540–547, 2004.

Aysel Ozgur, Pang-Ning Tan, and Vipin Kumar. Rba: An integrated framework for regression based on association rules. In *SDM*, 2004.

Feng Pan, Gao Cong, Anthony K. H. Tung, Jiong Yang, and Mohammed Javeed Zaki. Carpenter: finding closed patterns in long biological datasets. In *KDD*, pages 637–642, 2003.

Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.

Jian Pei, Jiawei Han, and Runying Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE*, pages 215–224, 2001.

Jian Pei, Jiawei Han, Hongjun Lu, Shojiro Nishio, Shiwei Tang, and Dongqing Yang. H-mine: Fast and space-preserving frequent pattern mining in large databases, 2004a.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng.*, 16(11):1424–1440, 2004b.

François Petitjean, Florent Masseglia, Pierre Gançarski, and Germain Forestier. Discovering significant evolution patterns from satellite image time series. *Int J Neural Syst*, 21(6):475–489, Dec 2011.

Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal. Multi-dimensional sequential pattern mining. In *CIKM*, pages 81–88, 2001.

Marc Plantevit, Yeow Wei Choong, Anne Laurent, Dominique Laurent, and Maguelonne Teisseire. M^2sp: Mining sequential patterns among several dimensions. In *PKDD*, pages 205–216, 2005.

Marc Plantevit, Anne Laurent, Dominique Laurent, Maguelonne Teisseire, and Yeow Wei Choong. Mining multidimensional and multilevel sequential patterns. *TKDD*, 4(1), 2010.

Uta Priss. Linguistic applications of formal concept analysis. In *Formal Concept Analysis*, pages 149–160. Springer, 2005.

Uta Priss. Formal concept analysis in information science. *ARIST*, 40(1):521–543, 2006.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012.

Chedy Raïssi and Jian Pei. Towards bounding sequential patterns. In *KDD*, pages 1379–1387, 2011.

Sherif Rashad, Mehmed M. Kantardzic, and Anup Kumar. Pac-whn: Predictive admission control for wireless heterogeneous networks. In *ISCC*, pages 139–144, 2007.

Toni M Rath and Raghavan Manmatha. Word image matching using dynamic time warping. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–521. IEEE, 2003.

Eric Sven Ristad and Peter N. Yianilos. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532, 1998.

Camille Roth, Sergei Obiedkov, and Derrick Kourie. Towards concise representation for taxonomies of epistemic communities. In *Concept Lattices and their Applications*, pages 240–255. Springer, 2008.

Anshuman Singh Sadh and Nitin Shukla. Apriori and ant colony optimization of association rules. *International Journal of Advanced Computer Research (IJACR) Volume-3 Number-2 Issue-10 June-2013*, 2013.

C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, 1(9):56–68, 1991.

Hassan Saneifar, Sandra Bringay, Anne Laurent, and Maguelonne Teisseire. S2mp: Similarity measure for sequential patterns. In *AusDM*, pages 95–104, 2008.

Lars Schmidt-Thieme. Algorithmic features of eclat. In *FIMI*, 2004.

Joan Serrà, Holger Kantz, Xavier Serra, and Ralph G. Andrzejak. Predictability of music descriptor time series and its application to cover song detection. *IEEE Transactions on Audio, Speech & Language Processing*, 20(2):514–525, 2012.

Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. pages 3–17, 1996.

Jerzy Stefanowski and Radoslaw Ziembinski. Mining context based sequential patterns. In *AWIC*, pages 401–407, 2005.

Gerd Stumme. Efficient data mining based on formal concept analysis. In *Database and Expert Systems Applications*, pages 534–546. Springer, 2002.

Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with titanic. *Data & knowledge engineering*, 42(2):189–222, 2002.

Thomas Tilley, Richard Cole, Peter Becker, and Peter Eklund. A survey of formal concept analysis support for software engineering activities. In *Formal concept analysis*, pages 250–271. Springer, 2005.

Romanas Tumasonis and Gintautas Dzemyda. The probabilistic algorithm for mining frequent sequences. In *ADBIS (Local Proceedings)*, 2004.

Romanas Tumasonis and Gintautas Dzemyda. Analysis of the statistical characteristics in mining of frequent sequences. In *Intelligent Information Systems*, pages 377–386, 2005.

Petko Valtchev, Rokia Missaoui, and Robert Godin. Formal concept analysis for knowledge discovery and data mining: The new challenges. In *Concept lattices*, pages 352–371. Springer, 2004.

Dean Van Der Merwe, Sergei Obiedkov, and Derrick Kourie. Addintent: A new incremental algorithm for constructing concept lattices. In *Concept Lattices*, pages 372–385. Springer, 2004.

Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos, and Eamonn J. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *KDD*, pages 216–225, 2003.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974. ISSN 0004-5411. doi: 10.1145/321796.321811. URL `http://doi.acm.org/10.1145/321796.321811`.

Hui Wang and Zhiwei Lin. A novel algorithm for counting all common subsequences. In *Proceedings of the 2007 IEEE International Conference on Granular Computing*, GRC '07, pages 502–, Washington, DC, USA, 2007. IEEE Computer Society.

Hui Wang, Zhiwei Lin, Sally McClean, and Jun Liu. Measuring similarity for multidimensional sequences. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 281–287. IEEE, 2010.

Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *ICDE*, pages 79–90, 2004.

Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: searching for the best strategies for mining frequent closed itemsets. In *KDD*, pages 236–245, 2003.

Jianyong Wang, Jiawei Han, and Chun Li. Frequent closed sequence mining without candidate maintenance. *IEEE Trans. Knowl. Data Eng.*, 19(8):1042–1056, 2007.

Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *Acm Sigkdd Explorations Newsletter*, 5(1):59–68, 2003.

Geoffrey I. Webb. Efficient search for association rules. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 99–107, New York, NY, USA, 2000. ACM. ISBN 1-58113-233-6. doi: 10.1145/347090.347112. URL `http://doi.acm.org/10.1145/347090.347112`.

Shlomo Weiss and Casimir Kulikowski. Computer systems that learn. 1991.

Rudolf Wille. *Restructuring lattice theory: an approach based on hierarchies of concepts*. Springer, 1982.

S.J. Wodak and J. Janin. Structural basis of macromolecular recognition. *Adv Protein Chem*, 61:9–73, 2002.

Xindong Wu, Chengqi Zhang, and Shichao Zhang. Efficient mining of both positive and negative association rules. *ACM Transactions on Information Systems (TOIS)*, 22(3):381–405, 2004.

Tengke Xiong, Shengrui Wang, Qingshan Jiang, and Joshua Zhexue Huang. A new markov model for clustering categorical sequences. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 854–863, Washington, DC, USA, 2011. IEEE Computer Society.

Jianhua Xu and Xuegong Zhang. Kernels based on weighted levenshtein distance. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 4, pages 3015–3018. IEEE, 2004.

Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large databases. In *SDM*, 2003.

Qiang Yang and Haining Henry Zhang. Web-log mining for predictive web caching. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):1050–1053, July 2003. ISSN 1041-4347.

Zhenglu Yang and Masaru Kitsuregawa. Lapin-spam: An improved algorithm for mining sequential pattern. In *ICDE Workshops*, page 1222, 2005.

Xiaoxin Yin and Jiawei Han. Cpar: Classification based on predictive association rules. In *SDM*, 2003.

Chung-Ching Yu and Yen-Liang Chen. Mining sequential patterns from multidimensional sequence data. *IEEE Transactions on Knowledge and Data Engineering*, 17:136–140, 2005. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2005.13.

Mohammed J. Zaki. Sequences mining in categorical domains: Incorporating constraints. In *9th ACM International Conference on Information and Knowledge Management*, Nov 2000a.

Mohammed J Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60, 2001.

Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12(3):372–390, 2000b.

Mohammed Javeed Zaki and Ching-Jiu Hsiao. Charm: An efficient algorithm for closed itemset mining. In *SDM*, 2002.

Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. In *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 283–286. AAAI Press, 1997.

Changhai Zhang, Kongfa Hu, Zhuxi Chen, Ling Chen, and Yisheng Dong. Approxmgmsp: A scalable method of mining approximate multidimensional sequential patterns on distributed system. In *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, volume 2, pages 730–734. IEEE, 2007.

Minghua Zhang, Ben Kao, Chi Lap Yip, and David Wai-Lok Cheung. Ffs - an i/o-efficient algorithm for mining frequent sequences. In *PAKDD*, pages 294–305, 2001.

Yanchang Zhao, Chengqi Zhang, and Longbing Cao. *Post-mining of association rules: techniques for effective knowledge extraction*. Information Science Reference-Imprint of: IGI Publishing, 2009.

Bo Zhou, David Wai-Lok Cheung, and Ben Kao. A fast algorithm for density-based clustering in large database. In *PAKDD*, pages 338–349, 1999.

Jianfei Zhu and Gösta Grahne. Reducing the main memory consumptions of fpmax* and fpclose. In *FIMI*, 2004.

Radosław Ziembinski. Algorithms for context based sequential pattern mining. *Fundamenta Informaticae*, 76(4):495–510, 2007.

Résumé

Tous les domaines de la science et de la technologie produisent de gros volume de données hétérogènes. L'exploration de tels volumes de données reste toujours un défi. Peu de travaux ciblent l'exploration et l'analyse de données séquentielles multidimensionnelles et hétérogènes. Dans ce travail, nous proposons une contribution à la découverte de connaissances dans les données séquentielles hétérogènes. Nous étudions trois axes de recherche différents: (i) l'extraction de motifs séquentiels, (ii) la classification et (iii) le clustering des données séquentielles. Tout d'abord, nous généralisons la notion de séquence multidimensionnelle en considérant la structure complexe et hétérogène. Nous présentons une nouvelle approche *MMISP* pour extraire des motifs séquentiels à partir de données séquentielles multidimensionnelles et hétérogènes. *MMISP* génère un grand nombre de motifs séquentiels comme cela est généralement le cas pour toues les algorithmes d'énumération des motifs. Pour surmonter ce problème, nous proposons une nouvelle façon de considérer les séquences multidimensionnelles hétérogènes en les associant à des structures de patrons. Nous développons une méthode pour énumérer seulement les motifs qui respectent certaines contraintes. La deuxième direction de recherche est la classification de séquences multidimensionnelles et hétérogènes. Nous utilisons l'analyse formelle de concept (AFC) comme une méthode de classification. Nous montrons l'intérêt des treillis de concepts et de l'indice de stabilité pour classer les séquences et pour choisir quelques groupes intéressants de séquences. La troisième direction de recherche dans cette thèse est préoccupé par le regroupement des données séquentielles multidimensionnelles et hétérogènes. Nous nous basons sur la notion de sous-séquences communes pour définir une mesure de similarité permettant d'evaluer la proximité entre deux séquences formées d'une liste d'ensemble d'items. Nous utilisons cette mesure de similarité pour construire une matrice de similarité entre les séquences et pour les segmenter en plusieurs groupes. Dans ce travail, nous présentons les résultats théoriques et un algorithme de programmation dynamique permettant de compter efficacement toutes les sous-séquences communes à deux séquences sans énumérer toutes les séquences. Le système résultant de cette recherches a été appliqué pour analyser et extraire les trajectoires de soins de santé des patients en cancérologie. Les données sont issues d' une base de données médico-administrative incluant des informations sur des patients hospitalisent en France. Le système permet d'identifier et de caractériser des épisodes de soins pour des ensembles spécifiques de patients. Les résultats ont été discutés et interprétés avec les experts du domaine.

Mots-clés: fouille de données, motifs séquentiels multidimensionnels, données hétérogènes séquentielles.

Abstract

All domains of science and technology produce large and heterogeneous data. Although a lot of work was done in this area, mining such data is still a challenge. No previous research work targets the mining of heterogeneous multidimensional sequential data. This thesis proposes a contribution to knowledge discovery in heterogeneous sequential data. We study three different research directions: (i) Extraction of sequential patterns, (ii) Classification and (iii) Clustering of sequential data. Firstly we generalize the notion of a multidimensional sequence by considering complex and heterogeneous sequential structure. We present a new approach called MMISP to extract sequential patterns from heterogeneous sequential data. MMISP generates a large number of sequential patterns as this is usually the case for pattern enumeration algorithms. To overcome this problem, we propose a novel way of considering heterogeneous multidimensional sequences by mapping them into pattern structures. We develop a framework for enumerating only patterns satisfying given constraints. The second research direction is in concern with the classification of heterogeneous multidimensional sequences. We use Formal Concept Analysis (FCA) as a classification method. We show interesting properties of concept lattices and of stability index to classify sequences into a concept lattice and to select some interesting groups of sequences. The third research direction in this thesis is in concern with the clustering of heterogeneous multidimensional sequential data. We focus on the notion of common subsequences to define similarity between a pair of sequences composed of a list of itemsets. We use this similarity measure to build a similarity matrix between sequences and to separate them in different groups. In this work, we present theoretical results and an efficient dynamic programming algorithm to count the number of common subsequences between two sequences without enumerating all subsequences. The system resulting from this research work was applied to analyze and mine patient healthcare trajectories in oncology. Data are taken from a medico-administrative database including all information about the hospitalizations of patients in Lorraine Region (France). The system allows to identify and characterize episodes of care for specific sets of patients. Results were discussed and validated with domain experts.

Keywords: data mining, multidimensional sequential patterns, heterogeneous sequential data.