Numéro d'ordre 2014-ISAL-0071 Année 2014

THÈSE

présentée devant
L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

pour obtenir
LE GRADE DE DOCTEUR

École doctorale : Mathématiques et Informatique (InfoMaths)

spécialité
INFORMATIQUE

par
ELISE DESMIER

CO-EVOLUTION PATTERN MINING IN DYNAMIC ATTRIBUTED GRAPHS

Soutenu publiquement le 15 juillet 2014 devant le jury composé de :

Pr. Jean-François Boulicaut	INSA de Lyon	Co-directeur
Pr. Bruno Crémilleux	Université de Caen Basse-Normandie	Rapporteur
Pr. Éric Gaussier	Université Joseph Fourier	Président
Pr. Donato Malerba	Università degli Studi di Bari Aldo Moro	Examinateur
Dr. Marc Plantevit	Université Claude Bernard, Lyon	Co-directeur
Pr. Pascal Poncelet	Université de Montpellier 2	Rapporteur
Pr. Céline Rouveirol	Université Paris-Nord	Examinateur

SUPERVISORS:
Jean-François Boulicaut
Marc Plantevit

Résumé

Cette thèse s'est déroulée dans le cadre du projet ANR FOSTER, "FOuille de données Spatio-Temporelles : application à la compréhension et à la surveillance de l'ERosion" (ANR-2010-COSI-012-02, 2011-2014). Dans ce contexte, nous nous sommes intéressés à la modélisation de données spatio-temporelles dans des graphes enrichis de sorte que des calculs de motifs sur de telles données permettent de formuler des hypothèses intéressantes sur les phénomènes à comprendre. Plus précisément, nous travaillons sur la fouille de motifs dans des graphes relationnels (chaque nœud est identifié de façon unique), attribués (chaque nœud du graphe est décrit par des attributs qui sont ici numériques), et dynamiques (les valeurs des attributs et les relations entre les nœuds peuvent évoluer dans le temps).

Nous proposons un nouveau domaine de motifs nommé motifs de co-évolution. Ce sont des triplets d'ensembles de nœuds, d'ensembles de pas de temps et d'ensembles d'attributs signés, c'est à dire des attributs associés à une tendance (croissance,décroissance). L'intérêt de ces motifs est de décrire un sous-ensemble des données qui possède un comportement spécifique et a priori intéressant pour conduire des analyses non triviales. Dans ce but, nous définissons deux types de contraintes, une contrainte sur la structure du graphe et une contrainte sur la co-évolution de la valeur des attributs portés par les nœuds. Prenons par exemple un graphe de co-publications scientifiques où les nœuds sont des auteurs reliés par une arête s'ils ont co-publié au moins un article au temps donné et où les attributs sont le nombre de publications dans un ensemble de conférences à chaque temps. Dans ce contexte, un motif peut être un ensemble d'auteurs qui ont co-publié ensemble sur plusieurs pas de temps et dont les publications ont augmenté sur ces pas de temps dans un sous-ensemble des conférences. Pour confirmer la spécificité du motif par rapport au reste des données, nous définissons trois mesures de densité qui tendent à répondre à trois questions. À quel point le comportement des nœuds en dehors du motif est similaire à celui des nœuds du motif ? Quel est le comportement du motif dans le temps, est-ce qu'il apparaît soudainement ? Est-ce que les nœuds du motif ont un comportement similaire seulement sur les attributs du motif ou aussi en dehors ? Considérant notre exemple, nous souhaitons extraire un ensemble d'auteurs dont les tendances de publication diffèrent des autres auteurs et sont spécifiques aux temps du motifs, et dont les tendances de publication diffèrent sur les autres conférences. Nous proposons l'utilisation d'une hiérarchie sur les attributs comme connaissance à priori de l'utilisateur afin d'obtenir des motifs plus généraux et adaptons l'ensemble des contraintes à l'utilisation de cette hiérarchie. Cette hiérarchie se traduit dans notre

exemple par des attributs nouveaux comme par exemple “conférences de fouille de données” et le motif peut alors porter sur une conférence précise ou sur la tendance de publication dans les conférences de fouille de données. Finalement, pour simplifier l’utilisation de l’algorithme par l’utilisateur en réduisant le nombre de seuils à fixer et pour extraire uniquement l’ensemble des motifs les plus intéressants, nous utilisons le concept de “skyline” réintroduit récemment dans le domaine de la fouille de données.

Nous proposons ainsi trois algorithmes MINTAG, H-MINTAG et Sky-H-MINTAG qui sont complets pour extraire l’ensemble de tous les motifs qui respectent les différentes contraintes. L’étude des propriétés des contraintes (anti-monotonie, monotonie/anti-monotonie par parties) nous permet de les pousser efficacement dans les algorithmes proposés et d’obtenir ainsi des extractions sur des données réelles dans des temps raisonnables. Afin de valider notre méthode, nous expérimentons sur plusieurs jeux de données (graphes) créés à partir de données réelles.

Abstract

This thesis was conducted within the project ANR FOSTER, “Spatio-Temporal Data Mining: application to the understanding and monitoring of erosion” (ANR-2010-COSI-012-02, 2011-2014). In this context, we are interested in the modeling of spatio- temporal data in enriched graphs so that computation of patterns on such data can be used to formulate interesting hypotheses about phenomena to understand. Specifically, we are working on pattern mining in relational graphs (each vertex is uniquely identified), attributed (each vertex of the graph is described by numerical attributes) and dynamic (attribute values and relations between vertices may change over time).

We propose a new pattern domain that has been called co-evolution patterns. These are trisets of vertices, times and signed attributes, i.e., attributes associated with a trend (increasing or decreasing). The interest of these patterns is to describe a subset of the data that has a specific behaviour and a priori interesting to conduct non-trivial analysis. For this purpose, we define two types of constraints, a constraint on the structure of the graph and a constraint on the co-evolution of the value worn by vertices attributes. Let us consider a graph of scientific co-publications where vertices are authors that are connected by an edge if they co-authored at least one paper at any given time and where the attributes are the number of publications in a set of conferences at every time. In this context, a pattern can be a set of co-authors who have published together on several times and whose publications have increased over the time steps in a subset of the conferences. To confirm the specificity of the pattern with regard to the rest of the data, we define three measures of density that tend to answer to three questions. How similar is the behaviour of the vertices outside the co-evolution pattern to the ones inside it? What is the behaviour of the pattern over time, does it appear suddenly? Does the vertices of the pattern behave similarly only on the attributes of the pattern or even outside? Considering our example, we want to extract a set of authors whose trends of publication differ from other authors and are specific to the times of the pattern, and whose trends differ on other conferences. We propose the use of a hierarchy of attributes as an a priori knowledge of the user to obtain more general patterns and we adapt the set of constraints to the use of this hierarchy. This hierarchy is reflected in our example by new attributes such as “data-mining conferences” and patterns can then focus on the publication trend on a specific conference or in data mining conferences. Finally, to simplify the use of the algorithm by the user by reducing the number of thresholds to be set and to extract only all the most interesting patterns, we use the concept of “skyline” reintroduced recently in the domain of data mining.

We propose three constraint-based algorithms, called MINTAG, H-MINTAG and Sky-H-MINTAG, that are complete to extract the set of all patterns that meet the different constraints. These algorithms are based on constraints, i.e., they use the anti-monotonicity and piecewise monotonicity/anti-monotonicity properties to prune the search space and make the computation feasible in practical contexts. To validate our method, we experiment on several sets of data (graphs) created from real-world data.

Publications

Publications related to the thesis

Most of the ideas presented in this thesis appear in the following publications:

International conferences (refered full papers)

DPRB13 Élise Desmier, Marc Plantevit, Céline Robardet, Jean-François Boulicaut. *Trend Mining in Dynamic Attributed Graphs.* Proc. European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases ECML PKDD 2013, Praha, Czech Republic, September 2013. Springer LNAI 8188, H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezný (Eds.), pp. 654-669.

DPRB12 Élise Desmier, Marc Plantevit, Céline Robardet, Jean-François Boulicaut. *Cohesive co-evolution patterns in dynamic attributed graphs.* Proc. 15th Int. Conf. on Discovery Science DS 2012, Lyon, France, October 2012. Springer LNAI 7569, J-G. Ganascia, P. Lenca, J-M. Petit (Eds). pp. 110-124.

National conferences (refered full papers)

DPB14 Élise Desmier, Marc Plantevit, Jean-François Boulicaut. *Granularité des motifs de co-variation dans des graphes attribués dynamiques.* Actes Extraction et Gestion des Connaissance EGC 2014, janvier 2014, Rennes (F), pp. 431-442.

Submission in preparation

DPRB14 Élise Desmier, Marc Plantevit, Céline Robardet, Jean-François Boulicaut. *Discovery of Skylines of Generalized Co-evolution Patterns in Dynamic Attributed Graphs.* To be submitted in May 2014 to the journal ACM Transactions on Knowledge Discovery in Databases.

Publications related to previous work (not discussed in the thesis)

Due to a previous internship in the PPME laboratory (University of New Caledonia), a member of the FOSTER project, the following papers have been published:

FNDS14 Frédéric Flouvat, Jean-François N'guyen Van Soc, Élise Desmier and Nazha Selmaoui-Folcher. Domain-driven co-location mining: Extraction, visualization and integration in a GIS. To appear in GeoInformatica, 2014, Springer.

DFGS11 Élise Desmier, Frédéric Flouvat, Dominique Gay, and Nazha Selmaoui-Folcher. *A clustering-based visualization of colocation patterns.* Proc. 15th International Database Engineering and Applications Symposium IDEAS 2011, Lisbon, Portugal, ACM, pp. 70-78.

DFSS11 Élise Desmier, Frédéric Flouvat, Benoit Stoll, Nazha Selmaoui-Folcher. *Coconut fields classification using data mining on a large database of high-resolution Ikonos images.* Proc. 6th IEEE Int. Conf. on Digital Information Management, ICDIM 2011, Melbourne, Australia, pp. 48-53.

Acknowledgments

I would first of all like to express my deepest appreciation to my advisors Marc Plantevit and Jean-François Boulicaut. Despite a difficult start, they have always pushed me and they have been able to learn me rigour of work I did not know being able to have. Working with them during this three years has taught me about data mining and research, but also about the world of work and about my own abilities. It was a great chance for me to have them as supervisors.

I would also thank all the members of the DM2L team. Especially Céline Robardet for our discussions, her advices and suggestions and to always be so positive, working with her has always been really pleasant. Mehdi Kaytoue and Keziban Gunce Orman for their support and joyfulness. Pierre-Nicolas Mougel, Thi Kim Ngan Nguyen and Adriana Bechara Prado for their welcome in the team and the time shared outside work. Christophe Rigotti for our long joyful journeys to the FOSTER meetings. I would also like to thank the members of the FOSTER project. Working with specialists in data mining, image processing and geologists has been a rewarding experience.

I would like to express my gratitude to Bruno Crémilleux and Pascal Poncelet for agreeing to be the reviewers of this thesis. I want to say a sincere thank to them for their time and interesting comments. And I would also like to express my gratitude to Éric Gaussier, Donato Malerba and Céline Rouveirol for agreeing to participate to the presentation.

I also want to thank all my friends that gave me courage during the difficult moments. These three years of my life in Lyon would not have been so pleasant without them. I would like to say a special thank to my boyfriend whose support during the difficult last months of the thesis was priceless and that helped me to give the best I could regardless the tiredness.

Most of all, I would like to thank my family for their support when I decided to begin this thesis and all along the three years. You have always been here, whatever the questions and the doubts, whatever your own difficulties. I am so proud and glad to be part of such a wonderful family.

I would like to dedicate this thesis to my beloved grandmother that I loved among all and that I will never forget.

Contents

Introduction

Graphs are a powerful mathematical abstraction that enables to naturally depict real-world phenomena. For instance, graphs provide a natural representation of important real life networks such as biological networks, social networks, spatio-temporal networks. Due to the success of social media, the growing of discovery in experimental sciences and the increasing access to data such as satellite positions, or captors information, such network data became increasingly available in the last decade. Consequently, network/graph mining is recognized as being one of the most studied and challenging tasks for the data mining community [26, 82]. A large part of the data are collected across time and these data are often spatially located whether they are spatial data as satellite images or a collection of sensors, or they are spatially connected as objects that are "connected by" for instance a road or moving objects that have been nearby. There is a lot of spatio-temporal data available and a need to understand them, i.e., understand the phenomena that appear in them [62, 68, 87]. That's why we decided to work on this type of data through the "FOSTER project" [1].

Traditionally, the focus on graph mining has always been done in two different and complementary ways. On the one hand, various approaches have been put forward to focus on macroscopic graph properties (e.g., degree distribution, diameter) [25] or graph partitioning [33]. On the other hand, many studies have been published on the extraction of more sophisticated properties within a pattern discovery setting. Computation of local patterns in graphs becomes quickly hard. For instance, one of the most frequent difficulty while studying graphs is the complexity of "simply" computing the number of isomorph graphs which belongs to NP [85] in the general case, NP is the set of decision problems solvable by a non-deterministic Turing machine that runs in polynomial time. Extracting local patterns in large datasets of any type, from collection of transactions (e.g., customer baskets) to more complex data representation such as graphs, is a highly studied algorithmic challenge. One of the most used method to answer this challenge is the use of constraint-based algorithms to reduce the search-space by not enumerating unpromising patterns. In [63], Mannila and Toivonen proposed the concept of theory; the aim is both to define the pattern language, i.e., the collection of patterns to compute, and to study efficient algorithms

[1]The FOSTER project ("Spatio-temporal data mining: application to the understanding and monitoring of soil erosion") is a three-year project launched in January 2011 and founded by the French National Research Agency ANR (ANR-2010-COSI-012-02)

to compute them. Given a dataset $\mathscr{D}$, a language $\mathscr{L}$ and a set of predicates $\mathscr{C} = \{c_1, ..., c_n\}$, the extraction task can be defined as: $\mathscr{T}(\mathscr{L}, \mathscr{D}, \mathscr{C}) = \{\phi \in \mathscr{L} | c_1(\mathscr{D}, \phi) \wedge ... \wedge c_n(\mathscr{D}, \phi) = \text{true}\}$. A constraint is a condition c_i that the solution ϕ must satisfy and a constraint satisfaction problem consists in finding all the solutions that satisfy a conjunction of constraints. The aim of constraints is to reduce the number of results by selecting the most interesting ones and the aim of constraint-based mining is to reduce the search space by using these constraints to early discard the unpromising candidates and then reduce the execution times. When defining a new pattern domain, the aim is then not only to define the pattern language but also to define a set of constraints associated to the pattern. One of the first type of pattern defined was the itemsets and one of the first constraint was frequency. The (anti-)monotonic property of the frequency over the specialization of an itemset has been successfully used by Agrawal et al. [3] when they defined the Apriori algorithm. This algorithm is one of the best known instance of the generic algorithm described in [63]. It allows to extract the complete set of frequent itemsets without enumerating a large part of the infrequent ones. Since then, both the pattern languages and the constraints evolved and both are becoming more complex. In particular, local pattern mining in graphs has been receiving much attention, leading to the introduction of new problems (like support counting in case of non-relational graphs [17, 20]) and resulting in new algorithms to mine collections of graphs [49, 74, 104, 106], single graphs [54, 80], and time-evolving graphs [8, 15, 86]. A majority of these methods use the concept of theory and define several constraints to improve the relevancy of the patterns. A pattern has to respect a conjunction of constraints and some of the defined measures used by the constraints can be complex. As the constraints become more complex, the (anti-)monotonicity property is no longer assured. New properties have been studied since then to help reduce the search space from (anti-)monotonicity to piecewise (anti-)monotonicity (also called primitive-based [95]) through convertible (anti-)monotonicity, loose (anti-)monotonicity etc. In the state of the art of his thesis [21], Cerf described all the classes of constraints that allow a reduction of the extraction times along with an increasing of the relevancy of the patterns.

In practice, many pieces of information are available and enable to focus on *vertex attribute augmented graphs* known as *attributed graphs* or *graphs with feature vectors* in the literature [65, 66, 84, 92]. Such attributed graphs enable to support more sophisticated discovery providing insightful patterns. For instance, let us illustrate this with the case of co-authorship networks which vertices naturally represent scientists and an edge appears between two scientists if they have co-authored at least one paper. It is the basic setting to highlight the main communities by focussing on the discovery of dense subgraphs [102]. Realistically, for each scientist (i.e. vertex), we know in which conference or journal she publishes and also her main research topics. As mentionned by Moser and Ester:,

> “often vertex attributes and edges contain complementary information, i.e. neither the relationships can be derived from the vertex attributes nor vice versa. In such scenarios the simultaneous use of both data types promises more meaningful and accurate results", [65].

Consequently, the consideration of vertex attributes enables to characterize group of collaborators with which domains (e.g., venues) the collaborations occur. Spatio-temporal phenomena can also be depicted easily as a dynamic attributed graph. The dynamic of the edges can represent some spatial relations or other types of interactions that change through time while the attributes allow to depict spatial or non-spatial information specific to the entity and that depend on time. Moreover, this type of graph allows to picture many type of data besides spatio-temporal ones as social data or biological data.

Attributed graphs are not the single case of augmented graphs. Indeed, evolving or dynamic graphs [8, 86, 107], multidimensional graphs [9] are also enriched with additional pieces of information. A growing body of literature has investigated augmented graphs but by only taking account one case. In this thesis, we proposed to tackle both dynamic and attributed graphs by introducing the problem of discovering co-evolution patterns in dynamic attributed graphs. This new kind of pattern relies on the graph structure and the temporal evolution of the vertex attribute values. The aim is to discover temporal phenomena in the data and we chose to find evolution of the attribute values, i.e., trends followed by the attributes over time, while keeping a certain consistence in the structure. More precisely we chose to find sets of vertices which follow the same trends on the values of a set of attributes over a set of times, and whose relations through the graph structure stay consistent. To be more robust towards intrinsic inter-individual variability, we do not compare raw numerical values, but their trends, that is, their derivative at timestamp t. To consider the consistence of the graph structure, we propose two methods:

- Cohesiveness. Pairs of vertices must have a similar neighbourhood, considering a threshold and a similarity measure, at each time of the pattern.
- Diameter. The connectivity of the dynamic subgraphs induced by the vertices and the times of the pattern is constrained by a maximum diameter value that limits the length of the longest shortest path between pairs of vertices.

Additional interestingness measures are used to assess the interest of the co-evolution patterns and guide their search with user-parametrized constraints. First, we propose some classic constraints based on the characteristics of the pattern itself, the size of the pattern and the maximality of the pattern, i.e., the existence or not of a pattern that provides strictly more information. All these constraints allow an increasing of the pertinence of the patterns, a reduction of the number of patterns and a reduction of the execution times using some of their properties. Second, we propose interestingness measures that compare the pattern to the rest of the dynamic attributed graph. This type of constraint is based on statistical measures. Here we propose measures to compare the behaviour of each set $s \in \mathscr{S} = \{vertices, times, attributes\}$ of the pattern on $\bar{s} = \mathscr{S} \setminus s$ compared to the behaviour of the rest of the dynamic attributed graph on $\bar{s}$. These measures aim at answering the following questions:

- How similar are the vertices outside the co-evolution pattern to the ones inside it?
- What about the dynamic of the pattern? Does it appear suddenly?
- Do the vertices of the pattern co-evolve only on the attributes of the pattern?

We go deeper in the analysis of dynamic attributed graphs by also taking into account the existence of a hierarchy over the vertex attributes where the "parent" attributes are abstraction of the attributes of the dataset. These attributes have no proper value but values computed thanks to a combination of their "children" attribute values. Indeed, we believe that the subsumption power of hierarchies can be useful to summarize patterns and avoid unperceptive/useless/meaningless patterns. Such taxonomies were already used in the extraction of sequences in [97] where the authors propose the extraction of generalized sequential patterns, i.e., they use taxonomies on the objects to extract sequences that are composed of multiple levels of hierarchy. Many studies have been done since then using such user knowledge information. We propose to mine hierarchical co-evolution patterns that satisfy some constraints on the graph topology and on the attribute values. The aim is to obtain patterns that provide information on a group of attributes, i.e., a set of entities whose behaviour mainly follows a trend on this group of attributes. We also propose two measures that answer the following questions:

- Is the pattern described with the good level of granularity?
- Does it need to be more specialized or generalized?

We propose two complete algorithms (MINTAG and H-MINTAG) to compute each type of pattern (with and without hierarchies) that traverse the lattice of (hierarchical) co-evolution patterns in a depth-first manner. It prunes and propagates constraints and thus takes advantage of a large variety of constraints that are usually not exploited by standard lattice-based approaches. However, all these propositions imply several threshold to be set, at least one for each constraint and some of these thresholds can be hard to fix for the user. A threshold too small can induce a huge set of patterns, and a threshold too high can induce no resulting pattern. Moreover, two constraints are not always compatible, for instance considering frequency and size, the more an itemset is large, the less frequent it is. We propose to use skyline analysis to extract patterns that are not dominated over a user defined subset of the proposed measures. The aim is to extract patterns that are not dominated by any other on a chosen set of measures, i.e., given a dominance relation on each measure, it does not exist any other pattern which dominates it on every measure[16, 80, 91, 96]. We design a new algorithm (Sky-H-MINTAG) where measures are used to assess the interest of the hierarchical co-evolution patterns and guide their search with user-parametrized constraints or within a skyline pattern analysis, i.e., interestingness measures as dimensions of the skyline analysis.

To summarize, our contributions are:

- The definition of co-evolution patterns, a new pattern domain for the study of dynamic attributed graphs. (This is mainly introduced in paper [28]).
- The introduction of interestingness measures, vertex specificity, temporal dynamic and trend relevancy. (This is mainly introduced in paper [29]).
- The definition of hierarchical co-evolution patterns and the introduction of the notions of purity and gain of purity. (This is mainly introduced in paper [27]).
- The introduction of the skyline operator that enables to extract patterns which are not dominated over a user chosen set of measures. (This is mainly introduced in LIRIS research report [30]).
- The design of three efficient algorithms, MINTAG, H-MINTAG and Sky-H-MINTAG that exploit the constraints, even those that are neither monotonic nor anti-monotonic.
- A quantitative and qualitative empirical study. We report on the evaluation of the efficiency and the effectiveness of the algorithm on several real-world dynamic attributed graphs, especially spatio-temporal ones.

Organisation of the manuscript

The manuscript is structured in three parts, a state of the art, the contributions and the experiments on real-world datasets.

Part I reviews a state of the art on pattern discovery in graphs. After an introduction on graph mining, we develop the survey of two types of graphs, attributed graphs and dynamic graphs and we discuss the study of dynamic attributed graphs.
Part II details the main contributions of this thesis. Chapter 2 defines the graph, the pattern language and some basic constraints. Two constraints on the structure of the graph are proposed and compared. An algorithm is presented with experiments on a real-world dataset to assess the effectiveness of the method. In chapter 3 we propose three improvements. First, some density measures to assess the specificity of the pattern considering the graph. Second, we introduce the use of a hierarchy on the attributes and adapt the pattern language. Last, we propose the use of skylines to stave off the difficulty of setting the thresholds of the constraints. For each improvement, we propose some experiments to evaluate the impact.
Part III present some applications[2]. First, we discuss the use of such a method on spatio-temporal data associated to the FOSTER project and we do preliminary experiments on one dataset

[2]Appendix A provides a detailed presentation of the datasets

extracted from satellite images. Second, we present results of experiments on four dataset created from a public database of domestic flights in the United States.
Finally, we conclude by giving a summary of the thesis.

Part I

State Of The Art And Preliminaries

Outline

Graph mining has become an extremely active research domain that has steadily increased [25]. The study of simple labelled graphs [47, 106], i.e., static graphs with only one type of edge and a simple label for each vertex, has recently been extended into several complementary directions as multidimensional graphs [10], attributed graphs [65, 66, 92], and dynamic graphs [15]. Indeed, entities can be described by one or more attributes that constitute the attribute vectors associated with the graph vertices. Moreover, in many applications, edges may appear or disappear through time giving rise to dynamic graphs. So far, the community has designed sophisticated methods to provide new insights from attributed or dynamic graphs. Recent contributions have shown that using additional information associated to vertices enables to exploit both the graph structure and local vertex attributes [65, 66, 92]. Dynamic graphs have been studied in two different ways. On one hand, it is possible to study the evolution of specific properties (e.g., the diameter). On the other hand, it makes sense to look at local patterns and it provides a large spectra of approaches to characterize the evolution of graphs with association rules [8, 73] or other types of patterns [15, 46, 50, 57, 86].

In this state of the art we present an overview of methods that locally study two types of graphs. Attributed graphs that allow to study both relations between entities and their specificity. Dynamic graphs that allow to study specific evolutions. We review the different types of methods and the different pattern domains. Finally we discuss the extension to dynamic attributed graphs and study three methods that work on such types of graphs, one using a clustering method, one that aim at characterizing communities and one extracting local evolving pattern.

1 — Pattern Discovery In Graphs

A simple graph $\mathscr{G}(\mathscr{V},\mathscr{E})$ is a set of vertices $\mathscr{V}$ where pairs are connected or not by edges $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$. A graph can easily represent entities and their relationships. Such types of graphs may be used to model biological data, as protein-protein interactions, or chemical data, as chemical compounds, for instance.

The main aim of studying simple graphs is to find subgraphs that occur frequently, i.e., set of entities that are typically structured in the same way in the dataset. In biological data, for instance, this can help to detect a behavior specific to a disease. The major difficulty of such frequent mining is the computation of subgraph isomorphism which is NP-complete in the general case [105] and several work have studied this problem [90, 109, 110]. Both [47] and [52] take advantage of an Apriori-based approach to discover all frequent subgraphs. In [106] the authors propose gSpan, an algorithm that adopts a depth-first search strategy. In [75], the authors propose the algorithm GASTON, that search frequent subgraphs with steps of an increasing difficulty, i.e., first frequent path, then frequent trees and finally frequent cyclic graphs. These works study collections of graphs, but some studies also aim at computing the support of a subgraph in a single graph, i.e., the number of isomorphism of a subgraph in a large graph [17, 20].

Another frequent study in simple graphs or collections of simple graphs mining is the discovery of cliques, i.e., set of vertices such that all vertices are pairwise connected in the graph[12, 36, 61, 100]. In the aim of finding dense but not necessarily complete subgraphs, pseudo-cliques[1] and quasi-cliques [49, 60, 92, 94, 108] have also been studied, i.e., cliques with a bounded number of allowed missing edges. Some work also study k-clique percolated components, i.e., cliques of size k sharing $k-1$ vertices that are merged to form larger connected patterns[2, 35, 66, 78, 79].

In this chapter, we review previous work in pattern mining in three types of graphs. Section 1.1 presents works on attributed graphs, i.e., graphs with attributes associated to the vertices, we study works on section dynamic graphs in Section 1.2 and Section 1.3 discusses recent works on dynamic attributed graphs.

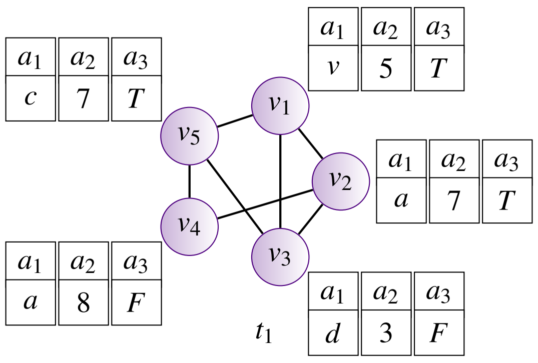
1.1 Pattern Mining In Attributed Graphs

An attributed graph is a simple graph with a set of attributes associated to the vertices. The edges can be directed or undirected. In a majority of the works studied here they are undirected but the transition to directed edges would be trivial. The set of attributes is common to all vertices, it is the value of the attribute that depends on the vertex. The set of attribute possible values is named its domain and it can be of any type, boolean, categorical, numerical.

Definition 1.1.1 An attributed graph is denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, f)$ with $\mathcal{V}$ a set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ a set of edges, $\mathcal{A} = \{a_1, \dots, a_n\}$ a set of attributes.
The function $f : \mathcal{V} \to dom(a_1) \times \cdots \times dom(a_n)$ associates a tuple of attribute values to each vertex $v \in \mathcal{V}$, with $dom(a_i)$ the domain of attribute a_i.

Attributed graphs allow to represent entities relations and specific characteristics. Even if it is similar, we differentiate attributed graphs from labelled graphs that only support one label on each graph. Labels could be treated as one categorical attribute or several binary attributes, indeed study of labelled and attributed graphs has similarities. For instance, in [101], the authors study labelled graph and extract exact matching or near matching subgraphs as an answer to a user query. They also discuss the possibility to adapt the method to the analysis of boolean attributed graphs. However, in this section we will only study attributed and not labelled graphs.

Boolean attributes allow to represent if an entity (vertex) is or is not associated to some characteristics (attributes). For instance, with a biological dataset where vertices are genes or chemical compounds, attributes can represent the drugs activating the genes or the compounds while the edges represent chemical reactions among the genes and the compounds.
Numerical or categorical attributes allow to characterize vertices over a set of common descriptors. For instance, with a scientific publications and cooperations dataset where vertices are scientific authors, attributes can represent the number of publications in a set of journals or its job position while there is an edge if two authors have worked together.
A toy example of an attributed graph is represented in Figure 1.1. In this example $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$, $\mathcal{A} = \{a_1, a_2, a_3\}$ and $dom(a_1) = \{a, c, d, v\}$, $dom(a_2) = \{3, 5, 7, 8\}$ and $dom(a_3) = \{T, F\}$, i.e. $\{True, False\}$.



a_1 a_2 a_3 / c 7 T
a_1 a_2 a_3 / v 5 T
a_1 a_2 a_3 / a 7 T
a_1 a_2 a_3 / a 8 F
a_1 a_2 a_3 / d 3 F
v_1
v_2
v_3
v_4
v_5
t_1

Figure 1.1: Example of an attributed graph.

Several approaches have been proposed to study local phenomena in this type of graphs. Many types of patterns are defined taking into account both attributes and the structure of the graph. Moreover they propose additional constraints to improve the quality of the patterns and reduce the number of patterns and the execution times. The most usual constraints are minimal support, that evaluate if the pattern appear frequently enough to be interesting, and the maximality, that only consider the pattern with the most information and avoid redundancy. Constraints more specific to the pattern domain are also defined, and the properties of all these constraints are used in the algorithms in order to reduce the search space. Indeed enumerating the totality of the possible pattern in such a type of graphs would be far too long.

Cohesive patterns

The work proposed in [65] is one of the first to treat at the same time the attributes and the structure of the graph. They propose to extract cohesive patterns, i.e., dense and connected subgraphs that have homogeneous values on their attributes. To this aim, they use a subspace cohesion function $s(V,T,\theta_s)$ to evaluate if the vertices of the graph are cohesive in a subspace of the attributes, which returns $True$ if and only if a set of vertices V associated to a set of attributes A has a homogeneity value greater than a threshold θ_s.

Definition 1.1.2 Given a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{A}, f)$, and three user-defined thresholds θ_s, θ_{dim} and α, a cohesive pattern is an induced subgraph $G = (V,E,A)$ s.t.:
- G is homogeneous in A, i.e., $s(V,A,\theta_s) = True$ and $|A| \geq \theta_{dim} \geq 1$,
- G is dense, i.e., $\frac{2\times|E|}{|V|\times(|V|-1)} \geq \alpha$,
- G is connected.

They define the cohesive pattern mining problem as finding the set of all maximal cohesive patterns of $\mathscr{G}$ w.r.t. θ_s, θ_{dim} and α. A cohesive pattern $G(V,E,A)$ is considered as maximal w.r.t. a graph $\mathscr{G}$ if it does not exist a set of vertex or a set of attributes that could be added to G without invalidating it. In social networks, for instance, such patterns can be used to discover communities with common interest (as the attributes). They propose CoPaM, a correct and complete algorithm to find all maximal cohesive patterns.

Proximity patterns

In [51], the authors present the problem of mining proximity patterns. A proximity pattern is a subset of attributes that repeatedly appear in multiple tightly connected subgraphs. In a social network with movies as attributes, a proximity pattern could represent a set of movies watched by multiple groups of users. The problem is related to frequent itemset mining, the itemset must however not appear entirely on a vertex (equivalent to a transaction) but on neighbors vertices. Two principles are taken from frequent itemset mining, the support and the downward closure. To adapt these principles to graph mining and proximity patterns they define an embedding of a set of labels I as the set of vertices that support the labels of I:

Definition 1.1.3 Given a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{A}, f)$ and a subset of vertices $V \subseteq \mathscr{V}$, let $f(V)$ be the set of attributes in V, i.e., $f(V) = \cup_{v \in V} f(v)$. Given an itemset $I \subseteq \mathscr{A}$ that is a subset of the attributes of the graph, V is called an embedding of I if $I \subseteq f(V)$.

They also define the strength of the embedding, i.e., how tightly the attributes are connected in the embedding. However computing the support while counting the number of embeddings implies two major issues. First, overlapped embeddings might be double counted, which violates the downward closure property. Second, embedding which are loosely connected and of negligible strength could be counted.

They propose two approaches to solve these issues: the neighbour association model and the information propagation model. The first one implies to generate all the embeddings of proximity patterns; however, if it solves the embedding overlapping issue, it is not feasible in practice. For the second one, the idea is to consider that the graph $\mathscr{G}$ represents a given timestamp, the information (attributes) is propagated through the graph until the graph reaches a stable state $\tilde{\mathscr{G}}$. Then the proximity patterns are frequent itemsets mined from $\tilde{\mathscr{G}}$. To compute the probability of propagation they study either the nearest probabilistic association (NPA) and the normalized probabilistic association (NmPA) to take into account all the nearest occurrences of a same attribute.

Itemset sharing subgraphs

In [89], the authors define itemset sharing subgraphs (ISS), i.e., (not necessarily dense) subgraphs with common itemsets. More formally:

Definition 1.1.4 Given a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{A}, f)$, an ISS is a connected subgraph $G = (V, E, I)$ of $\mathscr{G}$ where $V \subseteq \mathscr{V}$ is a set of vertices, $E \subseteq \mathscr{E}$ a set of edges and $I \subseteq \mathscr{A}$ is a set of items s.t. $I = I_{\mathscr{G}}(V)$, i.e., the set of common items of the vertices V in the graph $\mathscr{G}$.

The idea is then to extract the maximal ones to keep the ISSes of interest. The maximal ISSes are the largest ones, i.e., G where $\nexists$ an edge $(v_1, v_2) \in \mathscr{E}$ s.t. $v_1 \in V$ and $I_{\mathscr{G}}(V \cup v_2) = I$. Intuitively, an ISS is maximal if it is a subgraph that can not be enlarged with a connected vertex that is associated with at least the items of the pattern. In a social graph where attributes are purchased products, such a pattern could represent friends that purchased a same group of products, probably due to word-of-mouth communication. Instead of cohesive patterns presented previously, ISSes are not concerned in the density, which allows to extract sparser subgraphs. The authors design the algorithm COIN to compute the optimal solution.

In [34], they propose to extract ISS sets $\mathbb{G} = \{G_1, G_2, \ldots, G_n\}$ s.t. $I(\mathbb{G}) = \cap_{i=1..n} I_i$, where $V_i \cap V_j = \emptyset$, $\nexists G'$ s.t. $I' = I(\mathbb{G})$ except in $\mathbb{G}$ and $\nexists v$ in the neighbourhood of $G_i \in \mathbb{G}$ which is associated to the items of $I(\mathbb{G})$. Intuitively it is a set of non overlapping ISS that contains all the ISS associated to the attributes of the set and whose neighbourhood does not support these attributes. For instance, in the same social network, an ISS set could allow to identify sets of products that can be easily marketed through word-of-mouth communication. The authors also define 3 user-defined threshold θ_F, θ_I and θ_S s.t. the ISS set enumeration problem consist in enumerating all $\mathbb{G}$ that respect $|\mathbb{G}| \geq \theta_F$, $|I(\mathbb{G})| \geq \theta_I$ and $\forall G_i \in \mathbb{G}$ then $|G_i| \geq \theta_S$. Intuitively a set contains at least θ_F connected subgraphs with a size greater than θ_S and all the vertices of the set share at least θ_I attributes in common. To extract the ISS sets, the authors design the algorithm ROBIN that enumerate the ISSes and then combine them efficiently into ISS sets.

Attributed subtrees

In [81], the authors study a specific type of attributed graphs that are attributed trees, namely atrees. They propose to extract frequent attributed subtrees, namely asubtrees, i.e., isomorphic subtrees.

Definition 1.1.5 Given an atree $\mathscr{T} = (\mathscr{V}, \mathscr{E}, \mathscr{A}, f)$ where $(\mathscr{V}, \mathscr{E})$ is the underlying tree. An asubtree $T = (V, E, \lambda)$ of $\mathscr{T}$, denoted $T \sqsubset \mathscr{T}$ is an isomorphic subtree of $\mathscr{T}$, i.e., $\exists \phi : V \rightarrow \mathscr{V}$ s.t. $T \neq \mathscr{T}$, $(u, v) \in E$ if $(\phi(u), \phi(v)) \in \mathscr{E}$ and $\forall v \in V$, $\lambda(v) \subseteq f(\phi(v))$.

The authors then propose several properties to constrain the isomorphism, considering the preservation of the edges. The per-tree frequency is defined as the number of atrees that respect the isomorphism constraint and the problem statement is then to enumerate all frequent patterns in a given forest of atrees. In a real-world dataset such as trees that represent a user browsing on the web with web pages as vertices and keyword as attributes, such patterns could then represent browsing habits of users. The authors also design the algorithm IMIT to extract the frequent asubtrees in a collection of attributed trees.

Structural correlation patterns

In [93], the authors define structural correlation patterns. Instead of the previous presented patterns, this type of pattern is interested in finding attribute sets that explain the formation of dense subgraph through correlation. For instance, in a social network, the aim of the pattern can be to explain what communities emerge around a set of interests (attributes). A structural correlation pattern is a dense subgraph induced by a particular attribute set. More precisely it is a S-γ-quasi-clique, i.e., the induced subgraph is a dense component in the graph and there is a correlation among vertex attributes.

Definition 1.1.6 Given an attributes graph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{A}, f)$ a structural correlation pattern is a pair (V,S) where $S \subseteq \mathscr{A}$ is a set of attributes and V is a S-γ-quasi-clique s.t.

- $V \subseteq \mathscr{V}(S)$ where $\mathscr{V}(S) \subseteq \mathscr{V}$ is the set of vertices where each vertex has all attributes in S, i.e., $\forall v \in \mathscr{V}(S)$ then $f(v) \subseteq S$
- the graph induced by V is a γ-quasi-clique, i.e., $\forall v \in V$, v is connected to at least $\gamma \times (|V| - 1)$ vertices in V.

The authors also use constraints to improve the quality of the pattern and reduce the number of resulting patterns. The support of the attribute set, i.e., the size of $\mathscr{V}(S)$, must be greater than a given threshold and the considered γ-quasi-cliques must be larger than a minimum size. They also propose a clique membership function to evaluate how strongly the attribute set S is associated to the existence of γ-quasi clique in the graph. The authors design the algorithm SCORP that enumerates patterns in a depth-first search manner while using pruning strategies to not enumerate attribute sets and structural correlation patterns that are not promising. In [92], they improve the model to assess the significance/interestingness of a given structural correlation. To this aim, they propose to use normalized structural correlation which indicates how much the structural correlation of an attribute set S is higher than expected. The authors design the algorithm SCPM, which extends SCORP and extracts the top-k most relevant structural correlation patterns while using new pruning and search strategies.

K-clique percolated

In [66], the authors propose the task of finding collections of homogeneous k-clique percolated components (CoHoP). Shortly, a CoHoP is a set of vertices that share a set of attributes and whose relations form a relatively dense graph. For instance, in a social network, a CoHoP can represent a set of communities that share similar interests (attributes). This is close to the work of [34] that extract ISS sets but with a stronger constraint on the structure. To evaluate the density of the graph they choose to extract k-clique percolated components (k-PC), i.e., sets of cliques of size k connected by overlaps of $k-1$ vertices.

Definition 1.1.7 Given an attributed graph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{A}, f)$ and two user-defined thresholds α and δ, a CoHoP is a collection P of sets of vertices s.t.:

- the number of attributes shared by all sets of vertices is greater than a given threshold α, i.e., $A(P) = \cap_{V \in P}(\cap_{v \in V} f(v)) > \alpha$,
- P contains at least δ k-PC,
- P contains all k-PC that share the attributes of P, i.e., $A(P)$.

To reduce the collection of graphs and then improve the quality of the result they propose to extract only maximal k-clique, i.e., cliques of size at least k that are not a subset of any other clique. The authors also present an algorithm to mine the CoHoP and describe how to safely reduce the subgraphs enumeration.

Homogeneous clique sets

In [67], the authors define maximal homogeneous clique sets (HCSs), i.e., a subgraph structured in several large cliques not necessarily connected and where vertices share a sufficient set of labels. For instance, in a social network, a pattern could represent a set of persons with common interests (attributes), that do not necessarily know each others but with densely connected groups of friends. A HCS describes a subgraph organized around several dense and homogeneous set of vertices, it provides a different information on the structure and is complementary to [34] (ISS sets) and [66] (k-clique percolated).

Definition 1.1.8 Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, f)$ and three user-defined thresholds α, β and κ, an Homogeneous Clique Set (HCS) is a set $M = \{C_1, \dots, C_m\} \subseteq C(\mathcal{G})$, with $C(\mathcal{G})$ the set of cliques in $\mathcal{G}$ where:

- the vertices share at least α labels,
- M contains at least κ cliques of size at least β,
- the cliques in M are maximal in the subgraph induced by all the vertices appearing in M.

Then, they define a maximal homogeneous clique set (MHCS) which is a HCS φ that is maximal w.r.t. a defined partial order, i.e., if it does not exist another HCS φ that contains a super-clique of at least each clique in φ. The authors propose an algorithm correct and complete to extract all MHCSs using several pruning techniques to reduce the search space.

Subspace clustering

In paper [38], the authors propose a work at the interface of local and global analysis of the data, namely subspace clustering. The aim is to extract clusters of arbitrary shape and size while considering the attribute similarities in subspaces and the graph density. As for previous methods, in a social network, a pattern could represent close friends with similar interests. It differs from these methods on the structural constraint, indeed the constraint is more flexible than [89] or [66]. To take into account both graph topology and attributes, they define the combined neighbourhood $N^O(v)$ of a vertex v in a subset of vertices O and a subset of attributes S as the intersection of the structural neighbourhood and of the neighbourhood considering attributes, i.e., $N^{O,S}(v) = N_k^O(v) \cap N_{\varepsilon,S}^O(v)$. $N_k^O(v)$ returns all the vertices that are k-reachable from v in O. $N_{\varepsilon,S}^O(v)$ returns all the vertices whose attributes are at a distance lower than ε, i.e., all $u \in O$ s.t. $max_{i \in S}|u[i] - v[i]| \leq \varepsilon$.

Definition 1.1.9 Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \{$ and user-defined thresholds k, ε and $minPts$, a density-based combined subspace cluster $C(O,S)$ consists of a set of vertices $O \subseteq \mathcal{V}$ and a set of relevant attributes $S \subseteq \mathcal{A}$ s.t.:

- $\forall v \in O$, $|N^{O,S}(v)| \geq minPts$,
- $\mathcal{G}[O]$, the subgraph of $\mathcal{G}$ induced by O is connected,
- C is maximal, i.e., $\nexists O' \supset O$ that respects the previous constraints.

In order to keep the best non redundant clusters, they introduce an interestingness measure to $Q(C) = |O| \times |S|$ to optimize between the more vertices and the more attributes. Then, the considered clusters are those which are not redundant with another cluster with regard to their quality measure and their size. More formally, given two user-defined thresholds r_{obj} and r_{dim} a cluster $C(O,S)$ is considered as not redundant if $\nexists \overline{C}(\overline{O}, \overline{S})$ s.t. $Q(C) < Q(\overline{C}) \wedge \frac{|O \cap \overline{O}|}{|O|} \geq r_{obj} \wedge \frac{|S \cap \overline{S}|}{|S|} \geq r_{dim}$.

To discover the optimal clustering result with regard to the different threshold, they propose the algorithm DB-CSC.

Topological patterns

In [84], the authors propose to find co-variation among vertex descriptors. To this aim, they compute a set of topological properties which are then treated as additional attributes. They define topological pattern as a set of attributes associated to a trend, shortly a pattern is of the form for instance "the more a, the more b, the less c". For instance, in a social network where vertices represent authors and attributes represent the h-index, and the number of hours spent on instructional duties, a pattern could represent that authors that tend to have a high h-index, spend low time on instructional duties and publish with co-authors that are also central in the graph (represented by the topological measure of betweenness centrality). As for [51], this kind of pattern is more interested in the attributes, however it treats differently the graph structure as it does not only consider the neighbourhood of the vertices but also larger topological measures.

Definition 1.1.10 A topological pattern $P = (A_1^{s_1}, \dots, A_l^{s_l})$ is a set of vertex attributes and topological properties that behave similarly with A_i a vertex descriptor, i.e., either an attribute or a topological property, and $s_i \in \{+, -\}$ a trend.

They define the support of a topological pattern as the number of pairs of vertices that respect the trends of the patterns compared to the number of possible pairs of vertices. They also show that the support of a pattern is the same as the one of its symmetrical patterns (i.e., same attributes with the opposite trend) and that they are semantically equivalent, then, mining frequent topological patterns consists in extracting all frequent topological patterns but not their symmetric.

They also revisit the notion of emerging patterns to identify the most interesting topological patterns. To this aim, they propose two interestingness measures. First, they define a growth rate w.r.t. a selected attribute, the intuition is to identify the patterns that are mostly supported by pairs of vertices that also behave similarly on the selected attribute. Second they define a growth rate w.r.t. the graph structure, the intuition is to identify if the pairs of vertices that support the pattern are structurally correlated or on the contrary if the graph structure inhibit the support of the pattern. Finally, the authors design an algorithm, TopGraphMiner, that computes the frequent topological patterns and their top k representative vertices.

1.2 Pattern Mining In Dynamic Graphs

A dynamic graph is an ordered sequence of graphs directed or undirected with a conserved set of vertices. As for attributed graphs, the edges are mainly studied undirected but method could often be trivially adapted to directed edges.

Definition 1.2.1 A dynamic graph is a sequence $\mathscr{G} = (G_1, G_2, \dots, G_{|\mathscr{T}|})$ over a set of timestamps $\mathscr{T}$ of graphs $G_t = (\mathscr{V}, E_t)$ with $\mathscr{V}$ a constant set of vertices and $E_t \subseteq \mathscr{V} \times \mathscr{V}$ a set of edges depending on time.

A toy example of a dynamic graph is represented in Figure 1.2. In this example $\mathscr{V} = \{v_1, v_2, v_3, v_4, v_5\}$ and the edges set are equals to:

- $E_1 = \{(v_1,v_2),(v_1,v_3),(v_1,v_5),(v_2,v_3),(v_2,v_4),(v_3,v_5),(v_4,v_5)\}$,
- $E_2 = \{(v_1,v_2),(v_1,v_3),(v_2,v_3),(v_2,v_4),(v_4,v_5)\}$,
- $E_3 = \{(v_1,v_2),(v_1,v_3),(v_1,v_4),(v_2,v_3),(v_3,v_4),(v_4,v_5)\}$.

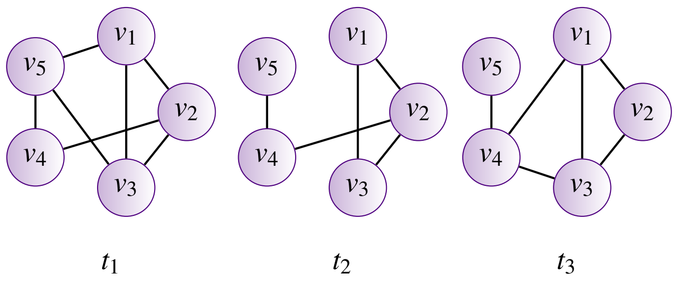


Figure 1.2: Example of a dynamic graph.

Dynamic graphs allow to represent evolution over time of the relations between vertices. We also present here papers that work on dynamic labelled graphs, i.e., sequences of labelled graphs. Many different approaches have been proposed to study this type of graphs. Some works study frequent subgraphs using isomorphism, i.e., they aim at extracting subgraphs that are frequent or that appear at a certain frequency in time. Some studies aim at discovering the evolution subgraphs through time, i.e., how the subgraphs structure change between consecutive timestamps. Some other studies aim at treating dynamic graphs as boolean tensors. Finally a study aim at localizing local congestion in networks.

Dynamic subgraphs

In [15], the authors propose to work on node-labelled dynamic graphs, i.e., time series of labelled graphs. They define topological subgraphs which are very close to the classic definition of subgraphs for static graphs, i.e., a subset of vertices that support a given label and a subset of edges that appear at the same timestamps. Then, they define dynamic subgraphs, intuitively a dynamic subgraph is a subsequence of topological subgraph.

Definition 1.2.2 Given a dynamic graph $\mathcal{G} = (G_1, G_2, \ldots, G_{|\mathcal{T}|})$ where $G_i = (\mathcal{V}, E_i, f)$ with $f : \mathcal{V} \rightarrow l \subseteq \mathcal{L}$ where $\mathcal{L}$ is a set of labels. A dynamic subgraph of lenght k $DSG = (G'_{start}, \ldots, G'_{start+k})$ is a subsequence of length k of a topological subgraph $DG = (G'_1, G'_2, \ldots, G'_{|\mathcal{T}|})$ with $G'_i = (V', E'_i, f)$ a subgraph of G_i.

The aim is then to find subgraphs that are topologically frequent and show an identical dynamic behaviour over time, i.e., appearance and disappearance of edges occur in the same order. For instance, in a social network where labels are the rank in the company and edges represent communications, a pattern could represent how the employees communicate considering their rank. The authors propose “Dynamic GREW” an extension of the algorithm of [53] to find all frequent topological subgraphs. An improvement is proposed in [103], where the authors design a more efficient algorithm.

Regular patterns

In [56], the authors look for regular patterns, i.e., patterns that occur at regular or near-regular time intervals, event if they are infrequent. For instance, in a dynamic graph representing the “social” interactions of animals, a pattern could describe seasonal association. An extended version with some improvements is proposed in [57]. To compute the support of a pattern, the authors define a periodic subgraph embedding (PSE), also named as periodic support set, as an ordered set of time steps separated by a constant number of timestamps.

Definition 1.2.3 Given a graph $\mathcal{G} = (G_1, G_2, \ldots, G_{|\mathcal{T}|})$ with $G_t = (\mathcal{V}, E_t, f)$. A periodic subgraph embedding (PSE) of a subgraph $G = (V, E)$ in $\mathcal{G}$ is a maximal orderd set of timesteps T where $\forall t \in T$, G is a subgraph of G_t and $\forall i = 1..|T|$ then $t_{i+1} - t_i = p$.

The aim is then to extract all the closed periodic subgraphs whose support is greater than a user defined threshold. They also propose some measures of purity that evaluate how much the relations of the patterns are respected outside the periodic timestamps. And they let the possibility to have near-constant period using either smoothing or jitter heuristic depending on the paper. They propose PSEMiner, an algorithm to extract all the parsimonious periodic subgraphs embedding, i.e., all PSE that are not subsumed by any other PSE (intuitively, all the maximal PSE).

Evolving patterns

In [86], the author proposes to extract evolving patterns over a time series of graphs. Dense subgraphs are extracted separately on each timestamp and associated to study their evolution. For instance in a dynamic social network, patterns could describe the evolution of groups of friends as the apparition of new persons in the groups. Subgraphs are considered as interesting if they are pseudo-clique, i.e., they have a high density of edges.

Definition 1.2.4 Given a user-threshold σ, a subgraph $G = (V, E)$ is considered as a valid pseudo-clique if $\frac{2 \times |E|}{|V| \times (|V|-1)} \geq \sigma$.

In the aim of studying the evolution of graphs, five temporal events are identified. Stability: the subgraph is a valid pseudo clique on two consecutive timestamps. Growth: the subgraph is a valid pseudo-clique and a subset of the subgraph was a valid pseudo-clique at the previous timestamp. Diminution which is the opposite of growth. Extinction: the subgraph was a valid pseudo-clique at previous timestamp but not any part of it is at the current timestamp. And emergence which is the opposite of extinction. A constraint-based algorithm, EVOLVING-SUBGRAPHS is defined to mine evolving patterns.

Rules to describe the graph evolution

In [107], the authors study how a graph is structurally transformed through time. To this aim, they compute graph rewriting rules that describe the evolution of two consecutive graphs. They define $S_{i,i+1}$, the maximal common subgraph between two consecutive graphs G_i and G_{i+1}.

Definition 1.2.5 Given a dynamic graph $DG = \{G_1, \ldots, G_n\}$ with $G_i = (V_i, E_i, L_{v_i}(V_i), L_{e_i}(E_i))$ a graph at time i. A graph rewriting rule is denoted $GR_{i,i+1} = \{(R_i, C_{R_i})(A_{i+1}, C_{A_{i+1}}$ where:

- $R_i = G_i \setminus S_{i,i+1}$ is the removal subgraph,
- $A_{i+1} = G_{i+1} \setminus S_{i,i+1}$ is the addition subgraph,
- C_{R_i} and $C_{A_{i+1}}$ are the set of connection edges for A_i and A_{i+1}.

These rules are then abstracted into patterns representing the dynamic of a sequence of graph, i.e., a pattern that compresses the learned graph rewriting rules to describe structural changes. For instance, in a social network where labels are the rank of the persons in a company and edges are the communications, a pattern could describe the changes in the relations. The authors propose an algorithm to discover rewriting rules whose main difficulty is the computation of the maximum common subgraphs between two consecutive graphs, which is NP-complete in the general case. However in the case of relational graphs (i.e. labelled graphs used in this method), the complexity is quadratic [17, 20] and rules are efficiently discovered.

In [44], the authors propose a method called GTRACE for mining frequent patterns from sequences of labelled graph. They propose to extract transformation rules, i.e., rules that represent graphs under the assumption that the change over consecutive graphs is gradual. The intuition of patterns extracted with this method is similar to the previous ones. A pattern is a sequence of graph $(g^1, \dots, g^n)$ where the change over adjacent graph g^i and g^{i+1} is gradual, i.e., only a small part of the graph change. They propose an improvement in [45], with GTRACE2, where the difference between two steps g^i and g^{i+1} may be a sequence of small changes they call intrastates.

Definition 1.2.6 An intrastate sequence $s^{(i)} = (g^{(j,1)}, \dots, g^{(j,m_i)})$ is a set of transformations between g^i and g^{i+1} where $g^i = g^{(j,1)}$ and $g^{i+1} = g^{(j,m_i)}$. Each transformation represents the insertion, deletion or relabelling of a vertex or an edge.

The aim is then to extract all the transformation sequences whose support is greater than a minimum support. Motivated by the possibility to mine patterns from graph sequences containing long sequences and large graphs, the same authors propose the mining of frequent relevant and induced subgraph subsequences in [46]. In this paper they propose to extract subgraph subsequences that are supported by a number of sequences greater than a user threshold and design a method called FRISSMiner to find frequent patterns in graph sequences.

In [8], the authors introduce the problem of extracting graph evolution rules in labelled evolving graphs. Evolving graph are defined as a simple graph with an attribute on the edge describing the first date appearance. First the method consists in finding frequent isomorphic subgraphs that are connected and whose support is greater than a user defined threshold. Then graph evolution rules are defined from these patterns that describe edges emerging in the future. For instance, in a social network where the edges represent co-authorship and the labels represent the degree of the node, a pattern that describes an highly labelled author connected to several authors with medium labels and which is later connected to a new author with a medium label could describe a local preferential attachment.

Definition 1.2.7 Given a pattern $P_H = (V_H, E_H)$, a rule if of the form $P_B \to P_H$ where $P_B = (V_B, E_B)$ such that:

- $E_B = e \in E_H | t(e) < max_{e' \in E_H} t(e')$ with $t(e)$ the attribute of the edge,
- $V_B = v \in V_H | \exists w \in V_H$ s.t. $(v, w) \in E_B$.

The support of the rule is defined as the support of P_H. And the authors propose two measure of confidence. They also design the algorithm GERM, which is an adaptation of the algorithm in [17] to mine evolution rules in labelled evolving graphs.

Dynamic graphs as Boolean tensors

In [24], the authors propose to treat dynamic graphs as ternary relations that describe the graph adjacency matrix at different timestamps. Then they define δ-contiguous closed 3-cliques as sets of vertices that stay densely connected on almost contiguous timestamps. Intuitively a pattern respects 3 constraints: it is a clique, i.e., there is an edge between each pair of vertices, it is nearly-contiguous, i.e., the clique is respected on almost consecutive timestamps, and it is closed, i.e., the pattern is maximal. More formally:

Definition 1.2.8 Given $a_{t,v^1,v^2} = 1$ if there is an edge between v^1 and v^2 at time t, a δ-contiguous closed 3-clique is a triset $P = (T, V^1, V^2)$ s.t.:
- P is connected and symmetric, i.e., $\forall (t, v^1, v^2) \in P$, $a_{t,v^1,v^2} = 1$ and $a_{t,v^2,v^1} = 1$,
- P is δ-contiguous, i.e., $\forall t \in T$, $\exists t' \in T$ s.t. $|t - t'| < \delta$,
- P is closed, i.e., $\nexists t \in \mathscr{T} \setminus T$ and $\nexists v \in \mathscr{V} \setminus (V^1 \cap V^2)$ s.t. $P \cup t$ or $P \cup v$ is connected.

For instance, in a social network that describe the interactions of participants in a conference, a pattern could represent temporary groups of work. The authors also describe how DATA-PEELER[22] can be specialized to support dynamic graphs while using the piecewise anti-monotonic properties of the constraints to reduce the search space.

In [73], the authors also treat dynamic graphs as boolean tensors that describe the graph adjacency matrix. However, the aim here is to extract multi-dimensional association rules that describe the graph evolution at a local level. For instance, in a social network that represent communications between employees of a company, a rule could describe that when two person a and b communicate, a communicates often few hours later with a person c. A multi-dimensional rule is of the form $X \rightarrow Y$ where $X \sqcup Y$ is an association rule. The frequency of the rule is defined as the proportion of elements in the domains outside the rule that support the rule. The confidence of the rule is not that straightforward, one must adapt the classic confidence measure to multi-dimensionality. The authors propose two measures:

Definition 1.2.9 Given a multi-dimensional rule $X \rightarrow Y$, a measure of confidence computes the proportion of elements in the dimension outside the rule that support $X \sqcup Y$ above those that support X.
- The exclusive confidence multiply the support of $X \sqcup Y$ by the number of elements in X and not in Y.
- The natural confidence replace the support of X by the support of X in the domain outside $X \sqcup Y$.

Intuitively, this allow these two measures to count the elements at the numerator and at the denominator on the same way, i.e., either on the dimensions outside $X \sqcup Y$ or on the dimensions outside X.

The authors also define non redundancy. Briefly speaking, that assesses that the rule is canonical, i.e., X and Y have no common element, and it does not exist another canonical rule such that it provides at least as much information with a smaller X. The aim is then to discover the complete set of non redundant rules that are frequent and confident enough. To this aim, the authors propose the algorithm PINARD++ that is an extension of GEAR[71] and PINARD[72].

High-scoring subgraphs

In [13], the authors aim at discovering the heaviest dynamic subgraph (HDS), i.e. the highest scoring temporal subgraph in a dynamic network with edge weight in $\{-1, 1\}$. Intuitively, an HDS is a connected set of active interactions that persist in time. In a communication network, for instance exchange of documents, such a pattern could represent a possible congested location over an extent in time.

Definition 1.2.10 Given an edge evolving network $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{F})$ with $\mathscr{F} = (f^1, \ldots, f^T)$ s.t. $f^i : E \rightarrow \{-1, 1\}$. A temporal subgraph $G = (V, E), [i, j]$ is a pair s.t. G is a connected subgraph and $[i, j]$ is a sub-interval of $[1, T]$.

The heaviest dynamic subgraph is then defined as the one that maximizes the score, i.e., the sum of all the edge-weight of the pattern. The authors propose the algorithm MEDEN to find the heaviest dynamic subgraph, that scales to large networks and long evolution extents.

Induced relational states

In [4], the authors propose to extract maximal non-redundant evolution paths among induced relational states (IRS). An induced relational set is a time-conserved set of relations among a fixed set of vertices. More formally, it is a tuple $S_i = (V_i, s_i : e_i)$ where V_i is the set of vertices of the induced subgraph that persists from snapshot G_{s_i} to G_{e_i}, i.e., where vertices are the same and edges are conserved on each timestamp. Then an evolving induced relational state identifies a sequence of time-persistent relational patterns. For instance in a social network, a pattern could describe the evolution of a group of friends, however, instead of the previous methods, evolving IRSs (EIRSs) identify stable groups of friends over a set of time and detect evolutions between these stable states.

Definition 1.2.11 Given a dynamic network $\mathscr{G}$ containing $|\mathscr{T}|$ snapshots, ϕ and β two user defined thresholds an EIRS of length m is a sequence of relational states $S = (S_1, \dots, S_m)$ s.t.:

- the supporting set of each S_i contains at least ϕ consecutive snapshots in the persistent dynamic network
- $\forall i \in [1..m]$, the first snaphshot in S_{i+1}'s supporting set follows the last one in S_i's supporting set
- given $G(S_i)$ the subgraph induced by S_i then $\forall i \in [1..m]$, $G(S_i) \neq G(S_{i+1})$
- $\forall i \in [1..m]$, $\frac{|V_i \cap V_{i+1}|}{|V_i \cup V_{i+1}|} \geq \beta$

They propose an algorithm to find all maximal EIRSs. In a similar way, the authors propose to study coevolving relational motifs (CRM), i.e., set of entities whose relations change in a consistent way over time, in [5], and propose the algorithm CRMminer.

Dynamic plane subgraphs

In [83], the authors aim at studying dynamic plane graphs. A plane graph is a labelled ordered graphs constructed by defining the list of neighbours of a vertex v in an anti-clockwise order around v.

Definition 1.2.12 Given a dynamic graph $\mathscr{G} = (G_1, \dots, G_n)$ and a plane graph P, the set of occurences of P in $\mathscr{G}$ is defined as $Occ(P) = \{(i, f) | f$ is an occurence of P in $G_i\}$. Where f is the corresponding injective function such that P is a subgraph isomorphism of G_i.

A spatio-temporal pattern based on P is then defined as the connected component of the occurrence graph of P where two occurences are connected if they are close. For instance, while representing a video as a series of plane graphs, patterns could correspond to objects that frequently appear in the video. The problem is then defined as computing all spatio-temporal patterns with a frequency greater than a user threshold, to this aim, the authors design the algorithm DyPlagram.
An extended version is proposed in [31] where the authors introduce additional spatio-temporal constraints and an extended version of the algorithm DyPlagram_ST that deals with the new constraints.

1.3 Discussion

All these methods treat graphs that are either associated to attributes sets or that are part of sequences. Many methods have been proposed to study these graphs, however, the way to represent real-world data is limited with such types of graph. Indeed, the attributes carry an important information additionally to the relations between vertices, but in real-world such information specific to the entities also evolve through time. The study of labelled dynamic graphs we presented in the previous section allows to describe pattern above the evolution of entities thanks to both their relation and their respective information. But labels allow a limited expression of entities specificity. That's why we are interested in the study of dynamic attributed graphs, i.e., sequences of graphs where vertex support a set of attribute values.

An example of dynamic attributed graph is presented in figure 1.3.

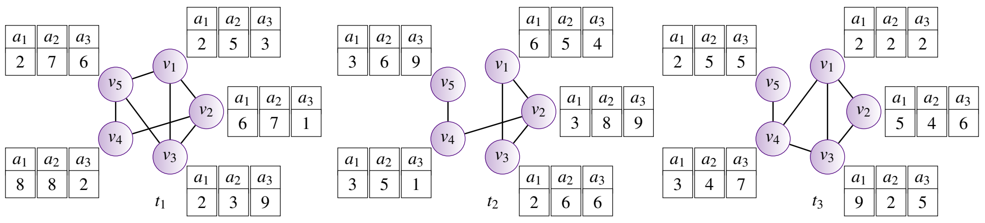


Figure 1.3: Example of a dynamic attributed graph.

To our knowledge, three papers treat dynamic attributed graph mining. Boden et al. [11], study sequences of attributed graphs in a global way. Indeed, they propose to trace clusters extracted in each static attributed graph over different time steps. First, they detect the clusters in the static graphs using the DB-CSC approach [38] presented in section 1.1. Then they associate time consecutive clusters that are similar. The authors of [76] propose a method to characterize communities in dynamic attributed networks by means of emerging sequential patterns. A pattern is a sequence of itemsets that is frequent within the community, i.e., that is supported by a sufficient proportion of the community.
In [50], Jin et al. consider dynamic graphs whose vertices are weighted, i.e., support one attribute, and introduce the Trend Motif approach which is a local pattern. Their objectives are quite similar to ours: they aim at analysing the dynamics of the network by discovering connected subgraphs whose vertex weights follow the same evolution. More precisely they extract groups of vertices whose weights follow a similar increasing or decreasing evolution, on a set of consecutive time stamps and whose induced subgraph is connected over this set of timestamps. For instance, in co-authorship network where the vertex attribute represents the number of publication in a conference, a pattern could represent a set of co-authors that increased their publication in this conference.

Definition 1.3.1 Given a graph $\mathcal{G}$, a trend motif is defined as the triset $(V_S, [t_s, t_e], f)$ where
- the induced subgraph $G[V_S]$ is a connected graph,
- $[t_s, t_e]$ is the maximal interval s.t. $t_s < t_e$ where the each vertex follow a same trend on its attribute value,
- f is a function $f : V_S \to \{+, -\}$ that associate a trend to the vertex attribute.

The problem is then defined as the extraction of frequent trends motifs, i.e., motifs that have a number of occurrence greater than a user threshold. The authors propose an algorithm that first extracts all the maximal intervals of trends to finally discover the trend motifs.

Aims of [50] are the most similar to the objectives of the approaches we propose in this thesis. Indeed, Jin et al. aim at discovering patterns that describe the (increasing/decreasing) trends followed over time by a set of connected vertices. We generalize this work in the sense that we focus on set of attributes while they focus on a unique attribute. Furthermore, the timestamps are not necessarily consecutive in our work. We also propose different constraints to take into account the graph structure, connectivity constraint being one specific case in our proposal.

Part II

Contributions

Outline

In this part of the manuscript, we introduce the problem of extracting co-evolution patterns in dynamic attributed graphs. I.e., a set of vertices which respects a constraint on the graph structure over a set of times and also a constraint on the trend of a set of attribute values over these times. Mining such patterns aims to extract entities that are related (expressed by the constraint on the graph structure) and whose characteristics co-evolve (expressed by the constraint on the attribute values).

We propose two ways to take into account the structure. The first method aims at comparing the neighbourhood of any pair of vertices to obtain a cohesive pattern. The second method aims at studying the connectivity of the induced subgraph to obtain patterns of a given maximal diameter. We compare these two methods to evaluate the relevancy on real-world datasets. In addition to this definition of co-evolution patterns and to allow the extraction of more relevant patterns, some additional constraints are proposed. Indeed, the collection of extracted patterns has to be "readable" and a too large set of patterns or too many similar patterns can lead to a diminution of the clarity of the collection. First, we define some basic constraints such as size constraints and maximality constraint. To discard patterns that do not bring a relevant information considering the dataset, we propose methods to compute the specificity of the pattern compared to the rest of the dynamic attributed graph. To reduce redundancy in the collection of patterns and obtain more general patterns, we propose the use of a hierarchy on the attributes. We define hierarchical co-evolution patterns that describe the trends of attributes or groups of attributes. The collection of extracted patterns is more readable, as a general pattern resumes the information of several specific patterns and the size of the collection is reduced. Finally, notice that there are as many thresholds than there are constraints which implies two main difficulties. First, for some of these constraints, the threshold can be hard to evaluate and may be set arbitrarily. Second, a pattern may be relevant even if it has not the best values on all measures. We propose the use of a skyline of hierarchical co-evolution patterns on a selected set of the measures to extract the most interesting patterns with less parameters to set.

To this aim, three constraint-based algorithms are proposed to extract the complete set of patterns in a reasonable time. All these propositions are tested on a real-world dataset which represents a set of co-authors and their number of publications in different conferences and journals.

2 — Mining Co-Evolution Patterns

The aim of co-evolution patterns is to depict a co-evolution of attribute values through time, i.e., similar trends. Several approaches are based on the correlation of attribute values. Hüllermeier [42] defined the extraction of "gradual" association rules of the form *"the more A, the more B"* using fuzzy partitions of the attribute domains instead of crisp partitions. This type of "gradual patterns" has also been applied to the study of spatio-temporal data in [40] to identify moving object clusters. In [32], the authors define closed frequent gradual itemsets and propose an efficient algorithm to mine large real-world datasets. Calders et al.[19] propose the extraction of rank-correlated sets of numerical attributes using support measures based on Kendall's tau, and Spearman's Footrule and rho instead of discretization. Recently, Appice et al. [6] proposed the discovery of trend clusters in spatial data streams, i.e., clusters of spatially close sensors whose variation of values is similar along the horizon time of a window. The authors of [84] define the problem of mining graph topological patterns, i.e., a set of vertex attributes and topological properties that behave similarly.

The chapter is organized as follows. In the next section we define the dynamic attributed graphs and the pattern domain. We propose a constraint on the trends of the attributes and two constraints on the structure and some basic additional constraints to obtain more relevant patterns considering the needs of the expert. In section 2.3, we devise an algorithm to compute the complete set of co-evolution patterns that respect the conjunction of constraints. Section 2.4 reports experimental results on a real-world dataset and discusses the results.

2.1 Dynamic Attributed Graphs

Dynamic attributed graphs allow to easily depict real phenomena where entities have particular characteristics and relations between each other. Entities are represented by the vertices, relations between entities are described by the edges and entities characteristics by the attribute values.

Definition 2.1.1 — Dynamic Attributed Graph. A dynamic attributed graph $\mathscr{G} = (\mathscr{V}, \mathscr{T}, \mathscr{A})$ is a sequence over a time period $\mathscr{T}$ of attributed graphs $\langle G_1, \cdots, G_{|\mathscr{T}|} \rangle$ where $\mathscr{A}$ is the set of attributes and each attributed graph G_t is a triplet $(\mathscr{V}, E_t, A_t)$, where:

- $\mathscr{V}$ is a set of vertices that is fixed throughout the time.

- $E_t \in \mathscr{V} \times \mathscr{V}$ is a set of edges at time t.
- A_t is a vector of numerical values for the attributes of $\mathscr{A}$ that depends on t.

Notation 2.1. *The value of the attribute α at time t for vertex v will be denoted $G_t(v,\alpha)$.*

An example of dynamic attributed graph is proposed as a toy dataset in Figure 2.1 with $\mathscr{V} = \{v_1, v_2, v_3, v_4, v_5\}$, $\mathscr{T} = \{t_1, t_2, t_3\}$ and $\mathscr{A} = \{a_1, a_2, a_3\}$. As an example $G_{t_1}(v_1, a_1) = 2$.

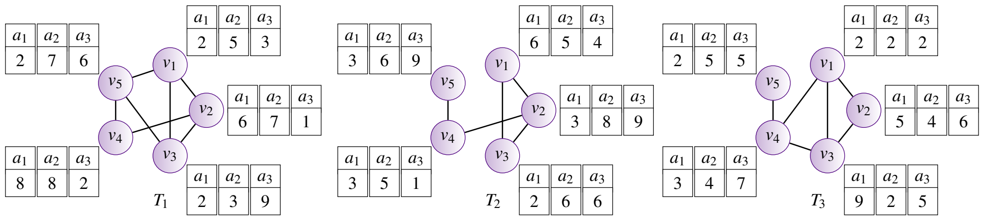


Figure 2.1: Toy example of a dynamic attributed graph.

The patterns studied in this manuscript are based on trends on the attributes, i.e., evolutions of the attribute values between two consecutive timestamps. That's why attributes of the graph must be ordinal, i.e., values of the attribute domains must be ordered. Then, in that case, the trend is defined by the order between the values observed at two consecutive timestamps. In the manuscript, the definitions are proposed based on the assumption that the attributes are numericals, but the adaptation to other ordinal attributes is straightforward, the domain of the ordinal attribute has just to be transformed by a bijective function in numerical ordered values. The representation of a dynamic attributed graph in this manuscript will be done like in Figure 2.2, i.e., the graph after pre-treatment and computation of the attribute trends. The edges of each time step t_i are the edges of the beginning timestamp T_i, and the trend is the evolution of the value between T_i and T_{i+1}. The number i of the time step t_i refers to the beginning timestamp T_i and there is $|\mathscr{T}| - 1$ time steps. This transformation of the values of the attributes as a trend can be done in pre-processing.

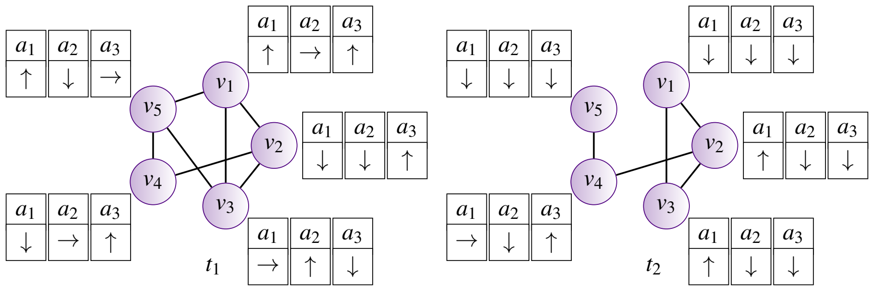


Figure 2.2: Toy example: from values to trends.

R To illustrate the definition of a dynamic attributed graph, let us present the dataset used in part II of this manuscript to evaluate and illustrate the methods and the algorithms. The graph is built from a subset of the digital library "DBLP" which is a computer science bibliography (`http://dblp.uni-trier.de/`). Vertices of the graph represent 2,145 authors who published at least 10 papers in a selection of 43 conferences and journals of the Data Mining and Databases communities between January 1990 and December 2012. This time period is divided in 10 timestamps, for sake of consistency in the data, we created overlapping periods. Each period describes the co-authorship relations and the publication records of the authors over 5 consecutive years, and two consecutive periods have a 3 years overlap ([1990-1994][1992-1996]...[2008-2012]). Each edge at a timestamp t links two authors who co-authored at least one paper in this time interval. Each vertex at each time is associated to a set of 43 attribute values corresponding to the number of publications in the 43 selected conferences and journals during the related period. To summarize, this dataset consists in 2,145 vertices, 10 timestamps and 43 attributes. This dataset is singular as each author has a significant scale of conferences/journals in which he has no publication all along its career. (More details on the "DBLP" dataset and a list of the selected conferences and journals can be found in Appendix A.)

As presented in Definition 2.1.1, the set of vertices and the set of attributes of a dynamic attributed graph are fixed. However, in real-world datasets, entities (represented as vertices in a dynamic attributed graph) and some of their characteristics (represented as attributes in a dynamic attributed graph) may appear or disappear through time. As an example in the "DBLP" dataset, a conference (attribute) can be created within the period. We choose to retain vertices that have no activity at a given date while representing them as isolated vertices where all attribute values are set to 0. During the time period where the vertex is absent, its attributes follow no evolution and it has no relation with its neighbours, then, no pattern can describe its co-evolution with related vertices. If there is no value for an attribute at time t, it is more difficult to treat. The attribute must be present and there must be a value for any vertex at time t, however filling the missing values without creating wrong patterns is really depending on the situation. As an example, for a conference created in 2000, there is no value before, however as no author published in it before, we can put value 0 for all authors at time before 2000 for this conference. Some patterns describing an increasing number of publications in it in 2000 could be extracted, but if the information is obvious it is valid. Moreover, if they depict the appearance of new vertices, these patterns are not obvious, that's why we choose not to set these attribute values to "NULL" and ignore them, but we choose to set values that do not invalidate the patterns. Instead, if there is a problem in the database and some missing values for a part or all authors at a given timestamp, filling them with either 0, the value of the previous timestamp or the next timestamp could impact the pattern extraction, as a decreasing or an increasing trend could be extracted. But, as the "constant evolution" is not treated in the trend analysis (the reason will be explain in Section 2.2), the treatment of missing values can be done in the computation of the trends, i.e., when passing from graph 2.1 to graph 2.2. A missing value at T_i can then be treated as a constant evolution in the time step before ,i.e., t_{i-1} and the time step after, i.e., t_i, then it will have a lower impact on the extraction of patterns.

The definition of a dynamic attributed graph specifies that the graph is non-oriented, however the use of oriented graphs would be trivial. Indeed, all the constraints proposed in this manuscript could be used with or adapted to oriented graphs.

2.2 Pattern Domain Definition

Intuitively, a co-evolution pattern is a set of vertices which are closely related through the graph structure and which follow the same trends over a set of attributes over time. Given a dynamic attributed graph $\mathscr{G}$, a co-evolution pattern $P = (V, T, \Omega)$ is a subset of $\mathscr{V}$, a subset of $\mathscr{T}$ and a subset of $\mathscr{A}$ which respect two constraints, one on the trend of the attribute values and one on the structure of the graph. As an example, on the DBLP dataset, a pattern is a set of authors that have similar co-authors and that have increased or decreased their number of publications in a subset of conferences or journals during a (possibly discontinuous) time period. We define two predicates:

- $related(P) = true$ denotes the fact that the structural relation is respected for all timestamps, i.e., $\forall t \in T$ then $related(V, E_t) = true$

- $evol(P) = true$ denotes the fact that the trend is respected for all triplet, i.e., $\forall v \in V, t \in T$ and $a^s \in \Omega$ the $evol(v,t,a^s) = true$. Where S is a set of trends. In this manuscript $S = \{+,-\}$ meaning respectfully a $\{increasing, decreasing\}$ trend.

The constraint $evol(v,t,a^s) = true$ can be any constraint on the evolution of the attribute values (in this manuscript, we define δ-$strictEvol$) and the constraint $related(V,E_t) = true$ can be any constraint on the evolution of the relations between vertices (in this manuscript, we define *cohesiveness* and *diameter*). Let us formalize co-evolution patterns:

Definition 2.2.1 — Co-evolution Patterns. Given a dynamic attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{T}, \mathcal{A})$, a co-evolution pattern, is a triplet $P = (V,T,\Omega)$:
- $V \subseteq \mathcal{V}$ is a subset of the vertices of the graph.
- $T \subset \mathcal{T}$ is a subset of not necessarily consecutive timestamps.
- Ω is a set of signed attributes, i.e., $\Omega \subseteq A \times S$ with $A \subseteq \mathcal{A}$.

The two following conditions must hold:
1. Each signed attribute a^s with $a \in A$ and $s \in \{+,-\}$ defines a trend that has to be satisfied by any vertex $v \in V$ at any timestamp $t \in T$: $evol(P) = true$
2. At each time $t \in T$, the vertices of the pattern have to be closely related through the structure E_t of the graph: $related(P) = true$.

Notation 2.2. *All along the manuscript, the set of attributes of the pattern will be referred as A and Ω indifferently when there is no ambiguity.*

R The set S could also be changed with a large variety of trends, as for instance $\{--,-,=,+,++\}$ with $--$ and $++$ meaning an evolution of more than 100% and $=$ meaning a value staying constant. There are several possibilities, however, if there are more than two trends, some constraints presented in chapter 3 should be redefined. In the early experiments of our work, we tried to use the trend $=$, however a lot of extracted patterns depicted constant evolution and brought no information. Indeed, in real-world datasets there is a majority of constant evolution, as the "DBLP" dataset that contains 95% of 0 on its attributes. Another example, in a dataset based on the weekly number of flights in airports and except events that will change radically the number of flights (e.g. extreme meteorological event), the number of flights is scheduled and stay mainly constant. The "Katrina" dataset, presented later (see appendix A for more information), contains 62% of constant trends.

2.2.1 The Evolution Constraint

As an instantiation of the predicate *evol* we propose to define a strict evolution constraint that compares the difference of values associated to an attribute between two consecutive timestamps. However, in many real-world applications, it is difficult to obtain strict equality of an attribute value between two consecutive timestamps, for instance when the values come from sensors. Thus, it is important to have the possibility to relax such a condition. To this end, we propose to consider a threshold δ that is a minimum amount of change, i.e., a vertex v validates the evolution (trend) s of an attribute a if its difference of value between times t and $t+1$ is greater than δ. Let us define formally the trend followed by a triplet (v,t,a):

Definition 2.2.2 — δ-trend(v,t,a). Given a threshold $\delta \in [0,1]$, $G_t(v,a)$ and $G_{t+1}(v,a)$ being two values of attribute a of vertex v at timestamp t and $t+1$. The trend of triplet (v,t,a) is defined as:

$$\begin{cases} \text{iff } (1+\delta) \times G_t(v,a) < G_{t+1}(v,a) & \Leftrightarrow \delta\text{-}trend(v,t,a) \quad = + \\ \text{iff } (1-\delta) \times G_t(v,a) > G_{t+1}(v,a) & \Leftrightarrow \delta\text{-}trend(v,t,a) \quad = - \\ \text{else} & \Leftrightarrow \delta\text{-}trend(v,t,a) \quad = \emptyset \end{cases} \tag{2.1}$$

Then, the constraint of evolution is defined as δ-strictEvol(P), with $P = (V, T, \Omega)$ the co-evolution pattern.

Definition 2.2.3 — δ-strictEvol(P). Given $P = (V, T, \Omega)$ and the function δ-$trend$, δ-strictEvol is valid iff:

$$\forall v \in V,\ \forall t \in T \text{ and } \forall a^s \in \Omega \text{ then } \delta\text{-}trend(v,t,a) = s \tag{2.2}$$

Most of the time, when studying datasets, we consider that the attribute value increases/decreases when its value changes, i.e., when $\delta = 0$. In this manuscript, we will refer mainly at $strictEvol(P)$ as an equivalent to 0-$strictEvol(P)$ and $trend(v,t,a)$ as an equivalent to 0-$trend(v,t,a)$.

■ **Example 2.1** Using $\mathcal{G}$ the graph in Figure 2.1 and the pattern $P = (\{v_1, v_2, v_3\}, \{t_2\}, \{a_2^{-\delta}, a_3^{-\delta}\})$.

- Given $\delta = 0$, $evol(P)$ is true as P is equivalent to $(\{v_1, v_2, v_3\}, \{t_2\}, \{a_2^-, a_3^-\})$
- Given $\delta = 0.5$, $evol(P)$ is false as $0.5 \times G_{t_2}(v_3, a_3) \leq G_{t_3}(v_3, a_3)$.
  However $P' = (\{v_1, v_2, v_3\}, \{t_2\}, \{a_2^{-\delta}\})$ respects $evol(P')$.

■

2.2.2 The Structural Constraint

It is now important to take into account the graph structure. A co-evolution pattern also depicts a set of vertices that are closely related through the graph. Indeed, the graph structure provides an additional information on the patterns, for instance, with the "DBLP" dataset, this relation allows to depict the working relations of the authors of the pattern. In this section two ways of considering the structure are presented. The first method makes sure that the neighbourhood of the vertices is cohesive, i.e., it computes the similarity of either the neighbourhood or the neighbourhood structure to find sets of close vertices. The second method makes sure that each pair of the set of vertices is connected by a path smaller than a maximum threshold in the induced subgraph (defined later).

Similarity of the vertex neighbourhoods

Let us now consider the cohesiveness of the pattern, i.e., the similarity of the neighbourhood in the dynamic attributed graph of pairs of pattern vertices. Given a minimum similarity threshold σ, for any two vertices $v, w \in V$, their similarity value with regard to a given similarity measure must be greater than σ.

Definition 2.2.4 — σ-cohesiveness(P). Given a similarity threshold $\sigma \in [0,1]$ and a similarity measure sim, a pattern $P = (V, T, \Omega)$ is said to be cohesive if the following condition holds:

$$\sigma\text{-}cohesive(V, T, \Omega) \equiv \forall t \in T,\ \forall u, v \in V^2,\ sim(u, v, G_t) \geq \sigma \tag{2.3}$$

The constraint of cohesiveness allows to check the similarity of vertices by pairs, i.e., the degree of likeness of their neighbourhood. Any similarity measure can be considered. Let us introduce some of them that exploit differently the graph structure. To simplify the notations, similarity measures are defined on a static graph $G(V, E)$ where the specific timestamp t is omitted. We can focus on the direct neighbourhood of vertices to assess how they are similar. To this end, cosine[98] or Jaccard[48] similarities are well adapted. Let $N(u)$ be the neighbourhood of u [1], i.e., $N(u) = \{v \in V | (u,v) \in E \vee v = u\}$ with $E \in V \times V$ the set of edges of the graph.

$$\begin{aligned} cosine(u,v) &= \frac{|(N(u)) \cap (N(v))|}{\sqrt{|N(u)| \times |N(v)|}} \\ Jaccard(u,v) &= \frac{|(N(u)) \cap (N(v))|}{|N(u) \cup N(v)|} \end{aligned}$$

These definitions require to take into account vertices that are adjacent to vertices u and v but also vertices u and v themselves to make it possible to have perfect similarity (i.e., similarity equals to 1) even if self loops are not allowed. Cosine and Jaccard measures only consider the direct neighbourhood, i.e., the adjacent vertices. However, one may exploit larger neighbourhood to establish similarity based on spreading activation. Intuitively, with spreading activation, the query nodes are activated, then activation is spread iteratively to adjacent nodes until a termination criterion is reached or the process converges. The result is a set of activated nodes and their level of activation. It enables to highlight vertices that play similar roles within the graph. Thiel and Berthold have introduced this kind of measure in [99]. They introduced two types of similarity both based on spreading activation. Thus the similarity between two vertices needs to consider and compare the graphs after diffusion of the activation with each of the two vertices as spreading starts.

$$\begin{aligned} a_v^0(u) &= 0, \text{if } v \neq u, \text{ 1 otherwise} \\ a_v^k(u) &= \frac{\sum_{(v,i)\in E} a_i^{k-1}(u)}{||\sum_{(v,i)\in E} a_i^{k-1}(u)||_2} \end{aligned}$$

The activation of the graph is associated to a beginning activation vertex u and is computed in k_{max} steps. $a^k(u)$ represents the vector of activation of the graph at step k of the spreading started at u and $a_v^k(u)$ represents the activation of vertex v, i.e., the value of the vertex v in the vector with $0 < k < k_{max}$. At Step 0, the only activated vertex is u with a value of 1. Thus, at the following steps $k \in [1, k_{max}]$, each vertex activation depends on its direct neighbour activation value at $k-1$.
Two similarity measures are now defined. The first one, called *the activation similarity*, compares the result of this spreading on the k-neighbourhood, and the common activated vertices.

$$\begin{aligned} \hat{a}^\star(u) &= D^{-1/2} \sum_{k=0}^{k_{max}} \alpha^k a^k(u) \\ Activation &= cos(\hat{a}^\star(u), \hat{a}^\star v)) \\ &= \frac{\sum_{i=1}^{n} \hat{a}_i^\star(u) \hat{a}_i^\star(v)}{||\hat{a}^\star(u)|| \, ||\hat{a}^\star(v)||} \end{aligned}$$

with α a decay parameter of a^k in the final result and D the degree matrix. The degree matrix is a diagonal matrix which contains the degree of each vertex in the graph. As the similarities are looking for common close neighbourhood, the spreading activation needs to loose weight over spreading with a certain value α. The more u and v have common highly activated vertices, the higher is the activation similarity $cos(\hat{a}^\star(u), \hat{a}^\star(v))$.

[1] By convention, u is in the neighbourhood of itself.

The second measure, called *the signature similarity*, compares how activation spreads into the neighbourhood and then compares the structure of the vertex neighbourhoods.

$$\begin{aligned}\delta^k(u) &= \begin{cases} 0, \text{ if } k=0 \\ a^k(u) - a^{k-1}(u) \end{cases} \\ \tau^k(u) &= ||\delta^k(u)||_2 \\ Signature &= cos(\tau(u),\tau(v)) = \frac{\sum_{k=1}^{kmax} ||\delta^k(u)||\,||\delta^k(v)||}{||\tau(u)||\,||\tau(v)||}\end{aligned}$$

The velocity vector $\delta^k(u)$ represents the change of activation of the graph between spreading steps and the convergence vector $\tau^k(u)$ describes the convergence speed in the spreading process. The more the structure of the vertex neighbourhoods is similar, the higher the signature similarity $cos(\tau(u),\tau(v))$. Thanks to these two similarities, we can compare a larger neighbourhood than with the cosine or Jaccard measures. Moreover, the signature similarity can be considered as a role comparison and it makes possible the extraction of completely different patterns.

■ **Example 2.2** Given $\sigma = 1$ and graph $\mathscr{G}$ as the one in Figure 2.2 (p. 44) and choosing *Jaccard* as similarity measure, let us study some examples:

- $P = (\{v_1, v_3\}, \{t_2\}, \{a_3^-\})$: $Jaccard(v_1, v_3) = 1$ at t_2. $related(P)$ is true and P is a co-evolution pattern. However, $P' = (\{v_1, v_3\}, \{t_1, t_2\}, \{a_3^-\})$ is false as even if $Jaccard(v_1, v_3) = 1$ at t_1 and $related(P')$ is true, $trend(v_1, t_1, a_3)$ is $+$, then $evol(P')$ is false.
- $P = (\{v_1, v_3, v_4\}, \{t_2\}, \{a_2^-\})$: $Jaccard(v_1, v_4) = \frac{1}{5}$ at t_2, so $related(P)$ is false.

■

Diameter of the induced subgraphs

Instead of computing a similarity on the neighbourhoods, let us now take into account the connectivity of the subgraph induced by the pattern. We consider to fix a maximal diameter, i.e., for any two vertices $v, w \in V$, there exists a path connecting them whose length is smaller than or equal to a given threshold at each time $t \in T$. Given $G_t(V) = (V, F_t)$ with $F_t = E_t \cap (V \times V)$ the sub-graph induced by V at time $t \in T$. Let $d_{G_t(V)}(v, w)$ be the shortest path length between the vertices v and w in $G_t(V)$, the diameter of G is thus defined by $diam_{G_t(V)} \equiv \max_{v,w \in V} d_{G_t(V)}(v, w)$. The constraint $diameter(P)$ is respected if $\forall t \in T$, the diameter of the graph $G_t(V)$ is less than or equal to a given threshold.

Definition 2.2.5 — Δ-diameter(P) or connectivity(P). Given $\Delta \in [1..|V|-1]$ a user-defined diameter threshold and $diam_{G_t(V)}$ the diameter of $G_t(V)$, the pattern P respects the diameter constraint if:

$$\Delta\text{-}diameter\big(V, T, \Omega\big) = true \Leftrightarrow \forall t \in T\; diam_{G_t(V)} \leq \Delta$$

If $\Delta = |\mathscr{V}| - 1$, the constraint $diameter(P)$ is equivalent to $connectivity(P)$, i.e., the induced sub-graph is connected, indeed it implies the subgraph induced by V is connected at each timestamp. Formally, $connectivity(P)$ is true if $(|\mathscr{V}| - 1) - diameter(P)$ is true.

Diameter allows to check how near are the vertices considering the graph structure. Thus, the value of the Δ threshold depends on the sparsity allowed to the subgraph induced by the pattern, see figure 2.3 to illustrate the impact of the threshold. $\Delta = 1$ implies that the subgraph is a clique, i.e., the vertices are highly connected. $\Delta = 2$ implies that the vertices of the subgraph have at least one common neighbour, some of them can then be coherent quasi-clique[49, 60, 108]. Then, the higher the Δ, the more the subgraph can be sparse and relations between vertices can be far. Until $\Delta = |\mathscr{V}| - 1$, then all the vertices are connected, but the subgraph can be a path. Considering the "DBLP" dataset, a small Δ allows to extract groups of authors that work together, as a high Δ allows to extract large groups of authors that attend to the same conferences or journals.

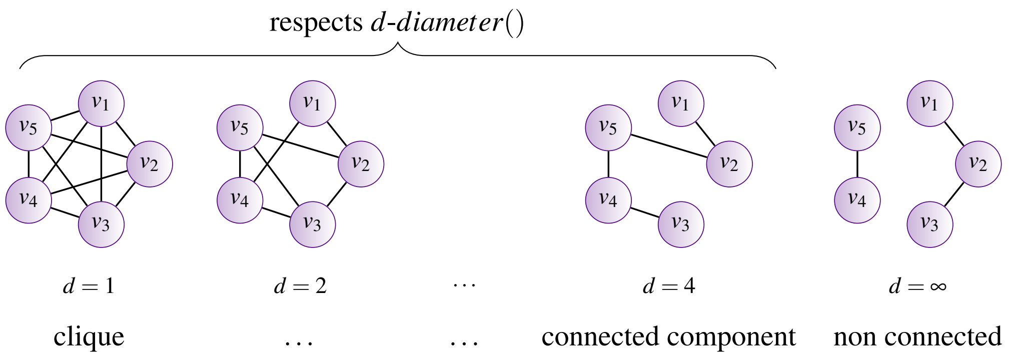


Figure 2.3: Illustration of the diameter impact

■ **Example 2.3** Given $\Delta = 2$ and graph $\mathscr{G}$ as the one in Figure 2.2 (p. 44), let us study some examples:

- $P = (\{v_1, v_2, v_3\}, \{t_2\}, \{a_3^-\})$: $diam_{(G_{t_2}(V))} = 1$. $related(P)$ is true and P is a co-evolution pattern. However, $P' = (\{v_1, v_2, v_3\}, \{t_1, t_2\}, \{a_3^-\})$ is false as even if $diam_{(G_{t_1}(V))} = 1$ and $related(P')$ is true, $trend(v_1, t_1, a_3)$ is $+$, then $evol(P')$ is false.
- $P = (\{v_1, v_2, v_4, v_5\}, \{t_1, t_2\}, \{a_2^-\})$: $diam_{(G_{t_2}(V))} = 3$, so $related(P)$ is false.

■

2.2.3 Additional Constraints To Improve The Relevancy Of The Patterns

The patterns defined by the constraints presented in Section 2.2 can be numerous and highly redundant.For instance using the "DBLP" dataset, all patterns with one author one timestamp and one trend could be extracted, which is not relevant. Moreover the expert does not have many ways to express his needs. That's why in this section, we propose additional constraints.

Constraint of maximality

A co-evolution pattern included in another pattern provides strictly less information, however these two patterns will be extracted as long as they respect co-evolution constraints. For instance, considering a new conference, many patterns will be extracted with an increasing of the number of publications in it. If authors a and b have co-published a paper in this new conference, a pattern with author a will be extracted together with a pattern with author b and a pattern with author a and b but the pattern with both authors is the most interesting.

Definition 2.2.6 — Order on the patterns. A pattern $P = (V, T, \Omega)$ is said to be a subset of another pattern $P' = (V', T', \Omega')$, denoted $P \subseteq P'$, if all of the elements of P are contained in P'. Formally, $P \subseteq P'$ iff $V \subseteq V'$, $T \subseteq T'$ and $\Omega \subseteq \Omega'$.

To avoid the extraction of such patterns, only maximal ones are extracted, i.e., patterns which are not a subset of another valid co-evolution pattern.

Definition 2.2.7 — maximal(P). P is maximal if it does not exist a pattern Q s.t. $P \subseteq Q$ and Q is a valid co-evolution pattern.

Some examples of trisets (V, T, Ω) are presented in Figure 2.4. The trisets are represented by the coloured elements, i.e., the triset on the left is $\{(v_2, v_3)(t_2)(a_3^-)\}$, the triset in the middle is $\{(v_1, v_2, v_3)(t_2)(a_2^- a_3^-)\}$ and the triset on the right is $\{(v_1, v_2, v_3, v_5)(t_2)(a_2^-, a_3^-)\}$. Considering

$\Delta = 2$, the two trisets on the left respect the diameter constraint but the one on the right does not as it is not connected and it can not be a co-evolution pattern. Then as the pattern on the left is included in the one in the middle it is not maximal. But the pattern in the middle is maximal, as even if it is included in the third triset, it is not included in any valid co-evolution pattern.

■ **Example 2.4** Given $\Delta = 2$ and $\mathscr{G}$ the graph in Figure 2.2 (p. 44).

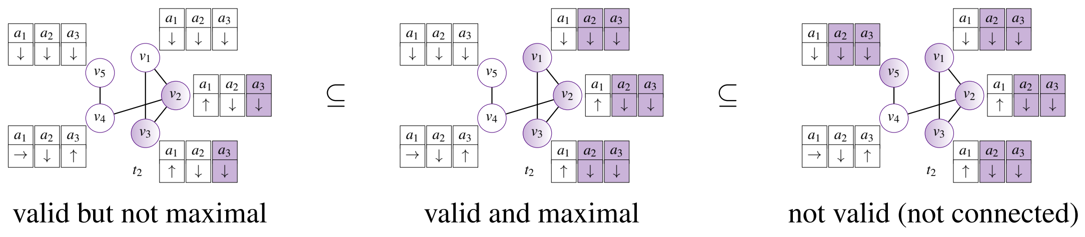


Figure 2.4: Example of relations of inclusion between the patterns and maximality of the patterns on 3 trisets. Trisets are represented by the coloured elements that have only one timestamp, i.e., $|T| = 1$.

■

Constraints of size

To avoid some patterns which could be useless considering the expert needs, some constraints are introduced to lower bound the size on the sets of the patterns. Often too small patterns are considered to be not relevant because they do not bring enough information. For instance a pattern with one vertex, one time and one signed attributed provides little information.

As simple constraints are often the useful ones, we first consider size measures that characterize a pattern by the number of elements it contains.

Definition 2.2.8 — Minimum size (min_V,min_T,min_A). The size measures consider the size of each set of the triplet (V,T,A) separately.

$$\begin{aligned} sizeV(P) &= |V| \\ sizeT(P) &= |T| \\ sizeA(P) &= |A| \end{aligned} \tag{2.4}$$

Given some user defined thresholds min_V, min_T and min_A, the constraints **minSizeV(P)**, **minSizeT(P)** and **minSizeA(P)** respectively imply $sizeV(P) > min_V$, $sizeT(P) > min_T$ and $sizeA(P) > min_A$. By definition $min_V > 0$, $min_T > 0$ and $min_A > 0$.

In some contexts, it can also be useful to combine the three size measures in a single time. For instance, a really small pattern is not interesting but only 2 vertices with many times and attributes or only one attribute with many vertices can be interesting. In these cases, size measures are of no use.

Definition 2.2.9 — Minimum volume (ϑ). The measure of volume evaluates the size as the three-dimensional volume of the pattern.

$$volume(P = (V,T,\Omega)) = |V| \times |T| \times |\Omega| \tag{2.5}$$

Given a user defined threshold ϑ, the constraint **volumeMin(P)** implies $volume(P) > \vartheta$. By definition $\vartheta > 0$.

■ **Example 2.5** Using $\mathscr{G}$ the graph in Figure 2.2 and the pattern $P = (\{v_1, v_2, v_3\}, \{t_2\}, \{a_2^-, a_3^-\})$.

- $sizeV(P) = 3$
- $sizeT(P) = 1$
- $sizeA(P) = 2$
- $volume(P) = 3 \times 1 \times 2 = 6$

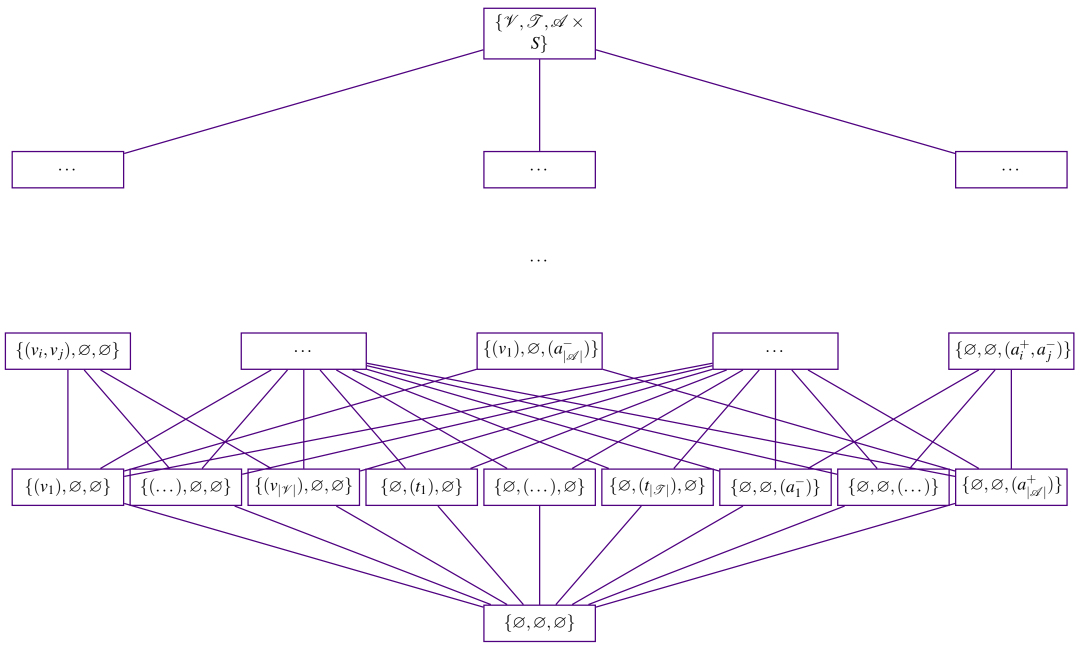
■

2.3 Algorithm

Given a dynamic attributed graph and a set of thresholds, the goal is to extract the complete set of co-evolution patterns satisfying the constraints. Formally, the collection of patterns $\mathscr{P} = \{P_1, \ldots, P_p\}$ such that P_i respects a conjunction of constraints $\mathscr{C} = \{evol, related, maximal, sizeMinV, sizeMinT, sizeMinA, volumeMin\}$. Only *evol*, *related* and *maximal* are mandatory constraints and only either σ or Δ has to be set as an input of the algorithm. The thresholds min_V, min_T, min_A and ϑ related to *sizeMinV*,*sizeMinT*,*sizeMinA* and *volumeMin* are optional and set to 1 by the algorithm if no value is assigned. Let us now present a complete algorithm under constraints to extract the collection of patterns $\mathscr{P}$ in a reasonable execution time.

2.3.1 General Structure Of The Algorithm

The search space of the algorithm can be depicted as a lattice presented in Figure 2.5 which contains all possible tri-sets from $\mathscr{V} \times \mathscr{T} \times (\mathscr{A} \times S)$, with bounds $\{\emptyset, \emptyset, \emptyset\}$ and $\{\mathscr{V}, \mathscr{T}, \mathscr{A} \times S\}$. The enumeration of all the patterns by materializing and traversing all possible tri-sets from the lattice is not feasible in practice. Therefore, we look for a decomposition of the original search space into smaller pieces such that each portion can be independently computed in main memory and such that the union of the co-evolution patterns extracted from each portion is the whole collection of co-evolution patterns. Then the constraints are used to reduce the search space while using their properties to not develop tri-sets that can not be valid co-evolution patterns. This enumeration strategy is the one proposed in [23] with slight adaptations.



$\{\mathscr{V}, \mathscr{T}, \mathscr{A} \times S\}$
…
…
…
…
$\{(v_i, v_j), \varnothing, \varnothing\}$
…
$\{(v_1), \varnothing, (a_{|\mathscr{A}|}^-)\}$
…
$\{\varnothing, \varnothing, (a_i^+, a_j^-)\}$
$\{(v_1), \varnothing, \varnothing\}$
$\{(\ldots), \varnothing, \varnothing\}$
$\{(v_{|\mathscr{V}|}), \varnothing, \varnothing\}$
$\{\varnothing, (t_1), \varnothing\}$
$\{\varnothing, (\ldots), \varnothing\}$
$\{\varnothing, (t_{|\mathscr{T}|}), \varnothing\}$
$\{\varnothing, \varnothing, (a_1^-)\}$
$\{\varnothing, \varnothing, (\ldots)\}$
$\{\varnothing, \varnothing, (a_{|\mathscr{A}|}^+)\}$
$\{\varnothing, \varnothing, \varnothing\}$

Figure 2.5: Part of lattice that represents the search space.

In the algorithm, all possible valid tri-sets are explored in a depth-first search manner. The enumeration can be represented as a tree where each node is a step of the enumeration. A node then contains 3 trisets denoted P, C and D:

- P is the pattern in construction and all the elements it contains will belong to any patterns generated from this branch of the enumeration tree,
- C contains the elements not yet enumerated and that can potentially be added to the pattern,
- D contains the elements that have already been enumerated as non member of the pattern under construction, D is used when checking the maximality.

At the beginning, P and D are empty and C contains all the elements of $\mathscr{G}$, i.e., $P = D = \varnothing$ and $C = \{\mathscr{V}, \mathscr{T}, \mathscr{A} \times S\}$.

R To avoid any misunderstanding, we use the term node for an element of the search space (i.e., a step of the enumeration containing P, C and D) and the term vertex to identify an entity of the dynamic attributed graph (i.e., an element $v \in \mathscr{V}$).

Notation 2.3. *For each triset, the set of vertices will be denoted $P.V$ (resp. $C.V$ and $D.V$), the set of times will be denoted $P.T$ (resp. $C.T$ and $D.T$) and the set of signed attributes will be denoted $P.\Omega$ (resp. $C.\Omega$ and $D.\Omega$).*

On the contrary of the algorithm proposed in [23], the enumeration used here is not binary, indeed, each node of the enumeration has not 2 children with or without an element to enumerate but $|C.\mathscr{E}| + 1$ children with $\mathscr{E} \in \{V, T, \Omega\}$ the element type that is enumerated. Therefore, at a node n, an element type $\mathscr{E}$ is selected and $|C.\mathscr{E}| + 1$ nodes are generated alternatively by

1. adding an element $e_i \in C.\mathscr{E}$ to P
2. adding elements $e_1..e_{i-1}$ to D
3. deleting elements $e_1..e_i$ from C

At last, a node is created by passing all the elements from $C.\mathscr{E}$ to $D.\mathscr{E}$ without adding any element to P. The process is recursively iterated until C becomes empty. This enumeration allows safe pruning of the search space thanks to the constraints of $\mathscr{C} = \{evol, related, maximal, sizeMinV, sizeMinT, sizeMinA, volumeMin\}$ and enables an algorithm correct and complete, i.e., that computes the complete set of co-evolution patterns and only valid patterns. Considering this enumeration, a majority of the constraints presented in section 2.2 (p. 45) are anti-monotonic, namely the constraints of evolution, of size, of volume and of cohesiveness. Connectivity and maximality are not, but they are partially monotone and an upper bound can be derived to prune parts of the search space. The general structure of the algorithm, i.e., without detail on the pruning techniques is presented in Algorithm 1. As will be detailed later, the tri-set D is only used to check maximality more easily.

Algorithm 1: MINTAG

```
Input: $P = \varnothing, C = (\mathscr{V}, \mathscr{T}, \mathscr{A} \times S), D = \varnothing, \mathscr{C}$
Output: Co-evolution patterns
begin
    if ($\mathscr{C}(P,C)$) then
        if $C = \varnothing$ then
            output $P$;
        else
            E ← ElementTypeToEnumerate($P$,$C$)
            for $i$ in $1..|C.E|$ do
                MINTAG($P \cup C.E[i]$,propagate($C \setminus C.E[1..i]$,$\mathscr{C}$),propagate($D \cup C.E[1..i-1]$,$\mathscr{C}$),$\mathscr{C}$)
```

Choice of the element type to enumerate

As we will see in the following section, the constraints make possible to reduce the search space by removing from C elements that can not form with P a valid pattern. However, each element has its own pruning power that depends on the current node configuration, therefore, the order in which elements are enumerated has a high impact on the efficiency of the algorithm. The *evol* constraint needs to have at least one vertex, one time and one attribute, and the *related* constraint needs to have at least one vertex and one time to prune the search space. Therefore, as a first step in the algorithm, one vertex, one time and then one attribute are enumerated. Moreover, as the patterns must have at least one of each ($min_V \geq 1$, $min_T \geq 1$ and $min_A \geq 1$), the enumeration node without any element of the chosen type is not enumerated, an example of the first steps of the enumeration without any pruning is presented in Figure 2.6. Once P contains one of each type, the element type to enumerate is chosen using the function proposed in [23]. The selected type is the $\mathscr{E} \in \{V,T,A\}$ that maximizes :

$$(C.\mathscr{E}' \times P.\mathscr{E}'') + (P.\mathscr{E}' \times C.\mathscr{E}'')$$

with $\mathscr{E}' \in (\{V,T,A\} \setminus \mathscr{E})$ and $\mathscr{E}'' \in (\{V,T,A\} \setminus (\mathscr{E} \cup \mathscr{E}'))$ being the two other element types. When the structural constraint used is the diameter, another slight adaptation is used when $\mathscr{E} = V$. Indeed, as the final induced subgraph must be connected, we only enumerate the vertices $v \in C.V$ that have at least one neighbour in $P.V$, i.e., $\exists u \in P.V$ such that $(u,v) \in E_t$, $\forall t \in P.T$.

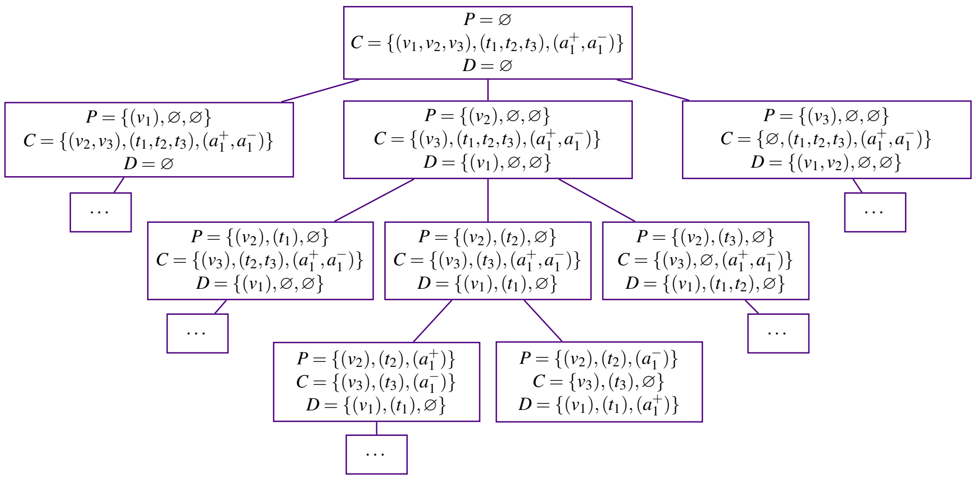

$P = \varnothing$
$C = \{(v_1,v_2,v_3),(t_1,t_2,t_3),(a_1^+,a_1^-)\}$
$D = \varnothing$
$P = \{(v_1),\varnothing,\varnothing\}$
$C = \{(v_2,v_3),(t_1,t_2,t_3),(a_1^+,a_1^-)\}$
$D = \varnothing$
$P = \{(v_2),\varnothing,\varnothing\}$
$C = \{(v_3),(t_1,t_2,t_3),(a_1^+,a_1^-)\}$
$D = \{(v_1),\varnothing,\varnothing\}$
$P = \{(v_3),\varnothing,\varnothing\}$
$C = \{\varnothing,(t_1,t_2,t_3),(a_1^+,a_1^-)\}$
$D = \{(v_1,v_2),\varnothing,\varnothing\}$
$\cdots$
$\cdots$
$P = \{(v_2),(t_1),\varnothing\}$
$C = \{(v_3),(t_2,t_3),(a_1^+,a_1^-)\}$
$D = \{(v_1),\varnothing,\varnothing\}$
$P = \{(v_2),(t_2),\varnothing\}$
$C = \{(v_3),(t_3),(a_1^+,a_1^-)\}$
$D = \{(v_1),(t_1),\varnothing\}$
$P = \{(v_2),(t_3),\varnothing\}$
$C = \{(v_3),\varnothing,(a_1^+,a_1^-)\}$
$D = \{(v_1),(t_1,t_2),\varnothing\}$
$\cdots$
$\cdots$
$P = \{(v_2),(t_2),(a_1^+)\}$
$C = \{(v_3),(t_3),(a_1^-)\}$
$D = \{(v_1),(t_1),\varnothing\}$
$P = \{(v_2),(t_2),(a_1^-)\}$
$C = \{v_3),(t_3),\varnothing\}$
$D = \{(v_1),(t_1),(a_1^+)\}$
$\cdots$

Figure 2.6: Example of the three first step of enumeration (without any pruning)

2.3.2 Properties Of The Constraints

Let us now present how the constraint properties are used to prune the search space.

Co-evolution

The strict co-evolution constraint (see definition 2.2.3, p. 47) is anti-monotonic, so it is easy to use it for pruning. First, if P does not respect $strictEvol(P)$, then no super-set of P can respect it, the branch of enumeration can be stopped. Second, it can be propagated to keep in C and D only elements that respect the evolution of P with regard to $strictEvol$. Indeed, the deleted elements would imply either useless enumeration nodes for C or a longer computation time for maximality for D. Obviously, if an attribute a^+ is added to P, then the attribute a^- is deleted from C or D.

Given $\mathscr{E} \in \{V, T, \Omega\}$:

$$C.\mathscr{E} \leftarrow \{e \in C.\mathscr{E} | strictEvol(P \cup e)\}$$
$$D.\mathscr{E} \leftarrow \{e \in D.\mathscr{E} | strictEvol(P \cup e)\}$$

Cohesiveness

The cohesiveness constraint (see definition 2.2.4, p. 47) is also anti-monotonic as similarity is computed between each pair of vertices. Then, given $P = (V, T, \Omega)$, this constraint can be propagated among C and D. Vertices v of C and D that do not respect $sim(P.V, v, G_{P.T}) > \sigma$ and times t of C and D when $sim(P.V, P.V, G_t) > \sigma$ is not respected can be deleted. It aims at avoiding useless enumeration nodes for C and reducing computation time for maximality for D. Indeed, the deleted elements would imply either useless enumeration nodes for C or a longer computation time for maximality for D.

$$C.V \leftarrow \{v \in C.V | \forall t \in P.T, \forall u \in P.V sim(u, v, G_t) > \sigma\}$$
$$C.T \leftarrow \{t \in C.T | \forall u, v \in P.V sim(u, v, G_t) > \sigma\}$$
$$D.V \leftarrow \{v \in D.V | \forall t \in P.T, \forall u \in P.V sim(u, v, G_t) > \sigma\}$$
$$D.T \leftarrow \{t \in D.T | \forall u, v \in P.V sim(u, v, G_t) > \sigma\}$$

Diameter and Connectivity

The diameter and connectivity constraint (see definition 2.2.5, p. 49) is neither monotonic nor anti-monotonic. Given $\Delta = 2$ and graph $\mathscr{G}$ as the one in Figure 2.2 (p. 44), the pattern $P = (\{v_1, v_4\}\{t_2\}\{\})$ does not respect $diameter(P)$ as v_1 and v_4 are not connected at time t_2. If the vertex v_2 is added, the resulting pattern $P' = (\{v_1, v_2, v_4\}\{t_2\}\{\})$ respects $diameter(P')$. However if the vertex v_5 is added, the resulting pattern $P'' = (\{v_1, v_2, v_4, v_5\}\{t_2\}\{\})$ does not respect $diameter(P'')$, indeed $d_{G_t(\{v_1,v_2,v_4,v_5\})}(v_1, v_5) = 3$ which is greater than Δ. Even if the constraint is not monotonic some properties of the constraint can be used to prune the search space.

First, the search space can be pruned thanks to P and C, if vertices of P can not respect Δ adding vertices of C. Formally, if $\exists t \in P.T$ s.t. $\max_{v,w \in P.V} d_{G_t(P.V \cup C.V)}(v, w) > \Delta$, then, the enumeration can be stopped as no valid pattern can be enumerated after this.

Second, the constraint can be propagated to C and D. Vertices v of C and D can be discarded if Δ can not be respected adding v to P, i.e., if $\exists w \in P.V$ and $\exists t \in P.T$ s.t. $d_{G_t(P.V \cup v \cup C.V)}(v, w) > \Delta$. And times t of C and D can be discarded if Δ can not be respected by P at this time, i.e., if $\exists v, w \in P.V$ s.t. $d_{G_t(P.V \cup C.V)}(v, w) > \Delta$. Formally, the elements kept in C and D are:

$$C.V \leftarrow \{v \in C.V | \max_{w \in P.V} d_{G_t(P.V \cup C.V)}(v, w) < \Delta\}$$
$$C.T \leftarrow \{t \in C.T | \max_{v,w \in P.V} d_{G_t(P.V \cup C.V)}(v, w) < \Delta\}$$
$$D.V \leftarrow \{v \in D.V | \max_{w \in P.V} d_{G_t(P.V \cup v \cup C.V)}(v, w) < \Delta\}$$
$$D.T \leftarrow \{t \in D.T | \max_{v,w \in P.V} d_{G_t(P.V \cup C.V)}(v, w) < \Delta\}$$

Size and Volume

The size constraints (see definition 2.2.8, p. 51) are anti-monotonic, so the branch of enumeration can stop if $|P.\mathscr{E} \cup C.\mathscr{E}| < min_{\mathscr{E}}$, with $\mathscr{E} \in \{V, T, A\}$. Indeed, as only elements of C can be added to P, the final pattern will never have a sufficient size.

The volume constraint (see definition 2.2.9, p. 51) is anti-monotonic and can be treated exactly as the size constraints. If the volume of P adding all candidates from C is not large enough to satisfy the constraint, the enumeration can be safely interrupted. More formally, if $volume(P \cup C) < \vartheta$, no valid pattern can be computed.

Maximality

Finally, checking the maximality constraint corresponds to making sure the conjunction of the *evol* and the *related* constraints is not valid for the patterns created adding an element of $\overline{P \cup C} = (\mathscr{V}, \mathscr{T}, \mathscr{A} \times S) \setminus (P \cup C)$ in $(P \cup C)$. If the resulting pattern respects all the constraints, then P can not be maximal. It's a complex constraint and can be really tough to compute as $\overline{P \cup C}$ can be really large. That's why the tri-set D is included in the algorithm. It contains all the elements that have been discarded from C but that still respect the constraints with P. Then to check maximality the only elements to consider are those of D and not those of $\overline{P \cup C}$, i.e., the only elements that could lead to a valid pattern. As the maximality is the conjunction of the preceding constraints, when using cohesiveness, maximality is anti-monotonic, but when using connectivity, maximality is neither monotonic nor anti-monotonic. Formally, given the *related* constraint as *cohesiveness* or *diameter* and considering checking diameter of a vertex v as verifying that the maximal distance of v with $P.V$, i.e., $\max_{w \in V} d_{G_t(P.V \cup C.V)}(v, w)$ is lower than Δ, the enumeration can be stopped if the following is not respected:

$$\begin{aligned} \forall v \in D.V,\ &\neg related(P \cup C \cup v) \wedge \neg evol(P \cup C \cup v) \\ \forall t \in D.T,\ &\neg related(P \cup C \cup t) \wedge \neg evol(P \cup C \cup t) \\ \forall a^s \in D.\Omega,\ &\neg evol(P \cup C \cup a^s) \end{aligned}$$

Framework of pruning and propagation

The order in which the constraints are presented is the order that must be used to do the pruning. Indeed, the constraint of co-evolution only needs the elements of P to prune C and D, then it must be used first. When using the constraint of cohesiveness it can also be used at the beginning. The constraint of diameter uses the elements of C to prune, then if used with the less possible elements in it, i.e., without those pruned with the co-evolution constraint, the pruning is more efficient. The constraints of size and volume need the elements of P and C and do not modify C and D, then they can be used at any moment but are more efficient if used after the *evol* and *related* constraints. At last, one can check the maximality. As for size and volume it does not modify C so it does not impact the efficiency of the pruning using *diameter* and it uses the elements of C and D so it must be checked after *evol* and *related* constraints (indifferently before or after volume and size, but as maximality is harder to compute, after is better).

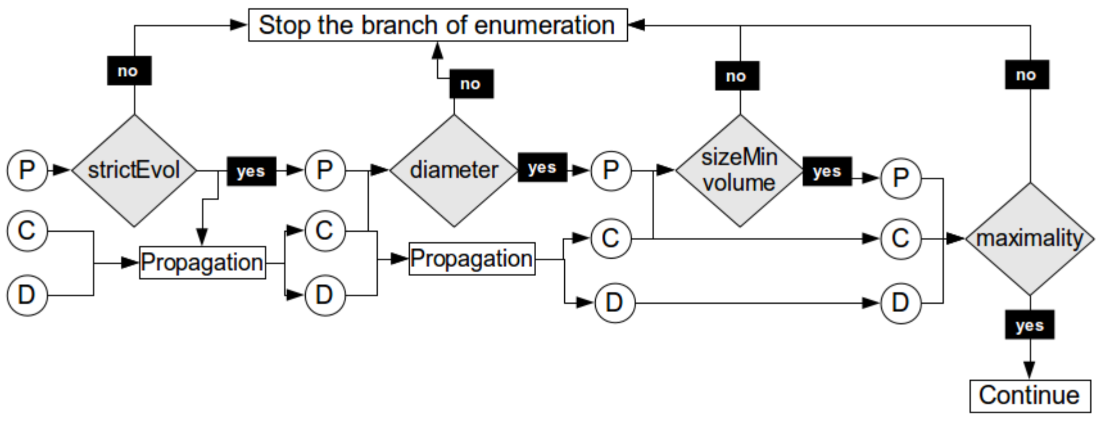
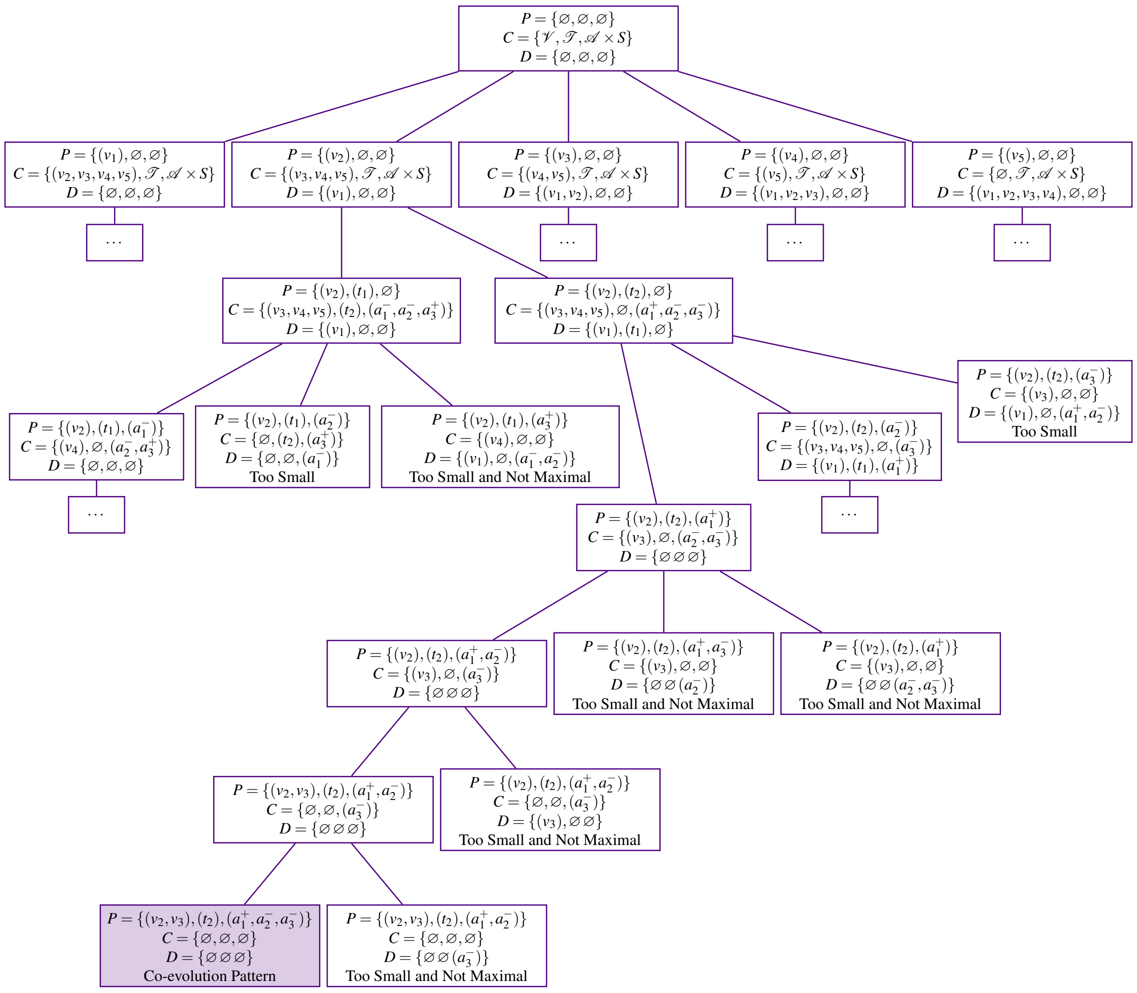


Figure 2.7: Framework of pruning and propagation.

2.3.3 Example Of An Execution Trace On A Toy Example

An example of an execution of the algorithm is proposed in Figure 2.8. The graph used is the toy example presented in figure 2.2 (p. 44), the constraint *diameter* is used with threshold $\Delta = 2$ and other thresholds are set to $\vartheta = 5$, $min_V = min_T = min_A = 1$. To keep the figure readable, only a part of the enumeration is presented, the other parts are represented by dots.


$P = \{\emptyset,\emptyset,\emptyset\}$ $C = \{\mathscr{V},\mathscr{T},\mathscr{A}\times S\}$ $D = \{\emptyset,\emptyset,\emptyset\}$
$P = \{(v_1),\emptyset,\emptyset\}$ $C = \{(v_2,v_3,v_4,v_5),\mathscr{T},\mathscr{A}\times S\}$ $D = \{\emptyset,\emptyset,\emptyset\}$
$P = \{(v_2),\emptyset,\emptyset\}$ $C = \{(v_3,v_4,v_5),\mathscr{T},\mathscr{A}\times S\}$ $D = \{(v_1),\emptyset,\emptyset\}$
$P = \{(v_3),\emptyset,\emptyset\}$ $C = \{(v_4,v_5),\mathscr{T},\mathscr{A}\times S\}$ $D = \{(v_1,v_2),\emptyset,\emptyset\}$
$P = \{(v_4),\emptyset,\emptyset\}$ $C = \{(v_5),\mathscr{T},\mathscr{A}\times S\}$ $D = \{(v_1,v_2,v_3),\emptyset,\emptyset\}$
$P = \{(v_5),\emptyset,\emptyset\}$ $C = \{\emptyset,\mathscr{T},\mathscr{A}\times S\}$ $D = \{(v_1,v_2,v_3,v_4),\emptyset,\emptyset\}$
…
…
…
…
$P = \{(v_2),(t_1),\emptyset\}$ $C = \{(v_3,v_4,v_5),(t_2),(a_1^-,a_2^-,a_3^+)\}$ $D = \{(v_1),\emptyset,\emptyset\}$
$P = \{(v_2),(t_2),\emptyset\}$ $C = \{(v_3,v_4,v_5),\emptyset,(a_1^+,a_2^-,a_3^-)\}$ $D = \{(v_1),(t_1),\emptyset\}$
$P = \{(v_2),(t_2),(a_3^-)\}$ $C = \{(v_3),\emptyset,\emptyset\}$ $D = \{(v_1),\emptyset,(a_1^+,a_2^-)\}$ Too Small
$P = \{(v_2),(t_1),(a_1^-)\}$ $C = \{(v_4),\emptyset,(a_2^-,a_3^+)\}$ $D = \{\emptyset,\emptyset,\emptyset\}$
$P = \{(v_2),(t_1),(a_2^-)\}$ $C = \{\emptyset,(t_2),(a_3^+)\}$ $D = \{\emptyset,\emptyset,(a_1^-)\}$ Too Small
$P = \{(v_2),(t_1),(a_3^+)\}$ $C = \{(v_4),\emptyset,\emptyset\}$ $D = \{(v_1),\emptyset,(a_1^-,a_2^-)\}$ Too Small and Not Maximal
$P = \{(v_2),(t_2),(a_2^-)\}$ $C = \{(v_3,v_4,v_5),\emptyset,(a_3^-)\}$ $D = \{(v_1),(t_1),(a_1^+)\}$
…
…
$P = \{(v_2),(t_2),(a_1^+)\}$ $C = \{(v_3),\emptyset,(a_2^-,a_3^-)\}$ $D = \{\emptyset\emptyset\emptyset\}$
$P = \{(v_2),(t_2),(a_1^+,a_2^-)\}$ $C = \{(v_3),\emptyset,(a_3^-)\}$ $D = \{\emptyset\emptyset\emptyset\}$
$P = \{(v_2),(t_2),(a_1^+,a_3^-)\}$ $C = \{(v_3),\emptyset,\emptyset\}$ $D = \{\emptyset\emptyset(a_2^-)\}$ Too Small and Not Maximal
$P = \{(v_2),(t_2),(a_1^+)\}$ $C = \{(v_3),\emptyset,\emptyset\}$ $D = \{\emptyset\emptyset(a_2^-,a_3^-)\}$ Too Small and Not Maximal
$P = \{(v_2,v_3),(t_2),(a_1^+,a_2^-)\}$ $C = \{\emptyset,\emptyset,(a_3^-)\}$ $D = \{\emptyset\emptyset\emptyset\}$
$P = \{(v_2),(t_2),(a_1^+,a_2^-)\}$ $C = \{\emptyset,\emptyset,(a_3^-)\}$ $D = \{(v_3),\emptyset\emptyset\}$ Too Small and Not Maximal
$P = \{(v_2,v_3),(t_2),(a_1^+,a_2^-,a_3^-)\}$ $C = \{\emptyset,\emptyset,\emptyset\}$ $D = \{\emptyset\emptyset\emptyset\}$ Co-evolution Pattern
$P = \{(v_2,v_3),(t_2),(a_1^+,a_2^-)\}$ $C = \{\emptyset,\emptyset,\emptyset\}$ $D = \{\emptyset\emptyset(a_3^-)\}$ Too Small and Not Maximal
Figure 2.8: Part of an execution of the algorithm.

Some nodes are noted “Too Small” because $volume(P \cup C) < 5$ and some are marked “Not Maximal” because one element of D is valid with elements of P and of C. For instance, the node at the last line but one on the right is both denoted “Too Small” and “Not Maximal”. It is considered too small as $volume(P \cup C) = (1+0) \times (1+0) \times (2+1) = 4$ which is lower than ϑ. And it is considered as not maximal as $trend(v_3,t_2,a_1) = +$, $trend(v_3,t_2,a_2) = -$, $trend(v_3,t_2,a_3) = -$ and $d_{G_{t_2}(P.V \cup C.V)}(v_2,v_3) = 1$ which is lower than Δ. Finally the coloured node is a valid co-evolution pattern as $strictEvol(P) = true$, $diameter(P) = true$, $C = \emptyset$ and $D = \emptyset$.

2.4 Empirical Study

Let us now present some experiments on the two types of patterns presented, i.e., using either the similarity constraint or the diameter constraint. All the experimental results presented in this manuscript were performed on a cluster. Nodes are equipped with 16 processors at 2.5GHz and 16GB of RAM under Linux operating systems. The algorithm has been implemented in C++ and compiled with GCC 4.1.2.

2.4.1 Co-Evolution Patterns With Similar Vertices

Let us now report on experimental results on the "DBLP" dataset using cohesiveness. This method is based on a similarity measure and we proposed four of them, cosine, Jaccard, activation and signature measures. These similarity measures have to be computed on each pair of vertices and their values do not change while varying the thresholds of the experimentations (i.e., σ, min_V, min_T, min_A, ϑ), then this can be treated as a pre-processing. Activation and signature similarities need two parameters. In [99], the authors used $\alpha = 0.3$ and they recommend not to have k higher than 10, then we choose $\alpha = 0.3$ and $k = 5$. The computation of these similarities on the "DBLP" dataset took from 28 seconds to 614 seconds. The distribution of similarities between vertices are reported in Figure 2.9. Each bar represents the number of couples whose similarity value is contained in the corresponding range of values of the horizontal axis. As the majority of the similarities are in a same range of values, the vertical axis of the figures is set logarithmic to be able to visualize the distribution of all the range of similarity values. The dataset contains 10 times, from t_0 to t_9. However the structure of the last time is not considered as vertices must respect the *related* constraint on the first time of the time-step (i.e., from t_0 to t_8) and respect the *evol* constraint between the two times of the time-step (i.e., from $t_0 - t_1$ to $t_8 - t_9$). Then the figures present distribution of similarities at first, middle and last time considered for the *related* constraint, i.e., t_0, t_4 and t_8.

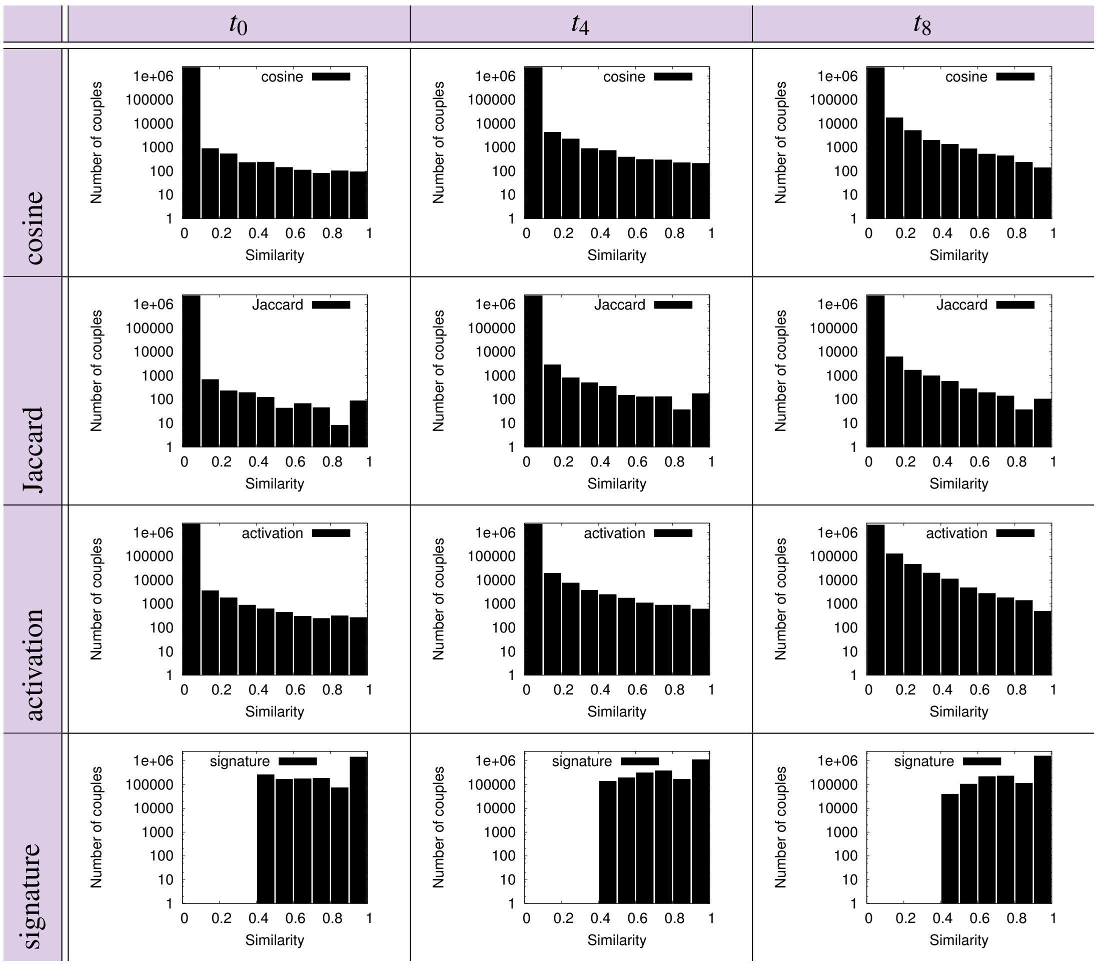


Figure 2.9: Distribution of the similarities between couples of vertices on the "DBLP" dataset (with exponential y axes).

Each measure has its own distribution, however cosine, Jaccard and activation similarities have a similar one. Almost all pairs of vertices have a similarity lower than 0.1, more than 98% for cosine and Jaccard and more than 90% for activation. Indeed activation similarity has more similarities distributed above 0.1 than cosine similarity that has more than Jaccard one. For these 3 similarities, we can see that the similarity between vertices increases over time, indeed, there is more pairs of vertices with a similarity higher than 0.1. Authors have more common co-authors around 2012 than around 1990, then they have more similar neighbourhood.

Studying the signature similarity, we can see that no couple has a similarity lower than 0.4 and more than 50% have a similarity higher than 90%. This can be explained by the fact that a lot of vertices are not connected to any other (see appendix A, p.131, for more details), due to no publication at certain years, before or after their career. All these authors have an identical neighbourhood and then a signature similarity equal to 1.

Regarding all these observations, we can conclude that a lot of couple of vertices are in a same range of values, whatever the similarity. This is helpful to fix the σ threshold.

Quantitative experiments

Figure 2.10 shows the execution times of our algorithm according to the volume threshold. Other parameters are set to $min_V = min_T = min_A = 1$ and $\sigma = 0.1$ using Jaccard similarity. The increasing of the volume threshold allows a drastic reduction of the number of patterns, passing from 37966 patterns with $\vartheta = 1$ to 1457 patterns with $\vartheta = 10$. In the same time, the running time decreases, meaning that the constraint enables to prune the search space. The volume allows to discard the smallest patterns which provide less information, and to reduce the number of extracted patterns and the execution times. The collection of patterns extracted is then easier to analyse.

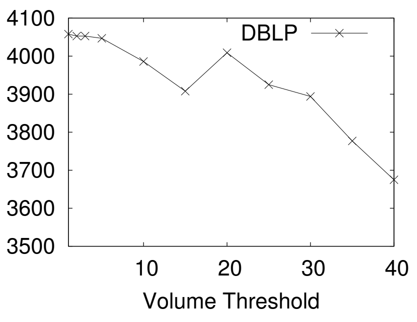
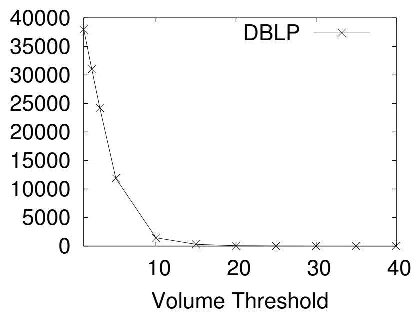
	Execution time (s)	Number of patterns
Volume		

Figure 2.10: Execution times and number of pattern extracted while varying the volume threshold (using Jaccard similarity and $\sigma = 0.1$).

Figure 2.11 presents the number of patterns and execution times with regard to the similarity threshold for each type of similarity except signature similarity. The volume threshold is set to $\vartheta = 20$ and the size thresholds are set to $min_V = min_T = min_A = 1$. Whatever the similarity type, the execution time explodes at a certain threshold. Indeed, the execution times are highly related to the distribution of vertex couples similarities and thus to the similarity threshold. For cosine similarity, the execution times explode around $\vartheta = 0.1$, for Jaccard similarity around $\vartheta = 0.22$, and for activation similarity around $\vartheta = 0.67$. Even if they have a similar distribution of similarities, the execution time increases more quickly, i.e., with a higher threshold, with the activation similarity. However the number of pattern extracted is in a same range of values for this 3 limits, respectively 49, 59 and 40. Moreover the execution time increases less exponentially using the activation similarity as there is more couple similarities distributed between 0.1 and 1.

For signature similarity, no results are presented as the algorithm does not scale on this dataset using this similarity. Indeed, there is a high number of couples having a similarity of 1, then even with $\sigma = 1$ the constraint is not enough stringent to reduce the search space and lead to an acceptable execution time. The algorithm did scale on a smaller synthetic dataset (see paper [28] for more details), however it also exploded quickly.

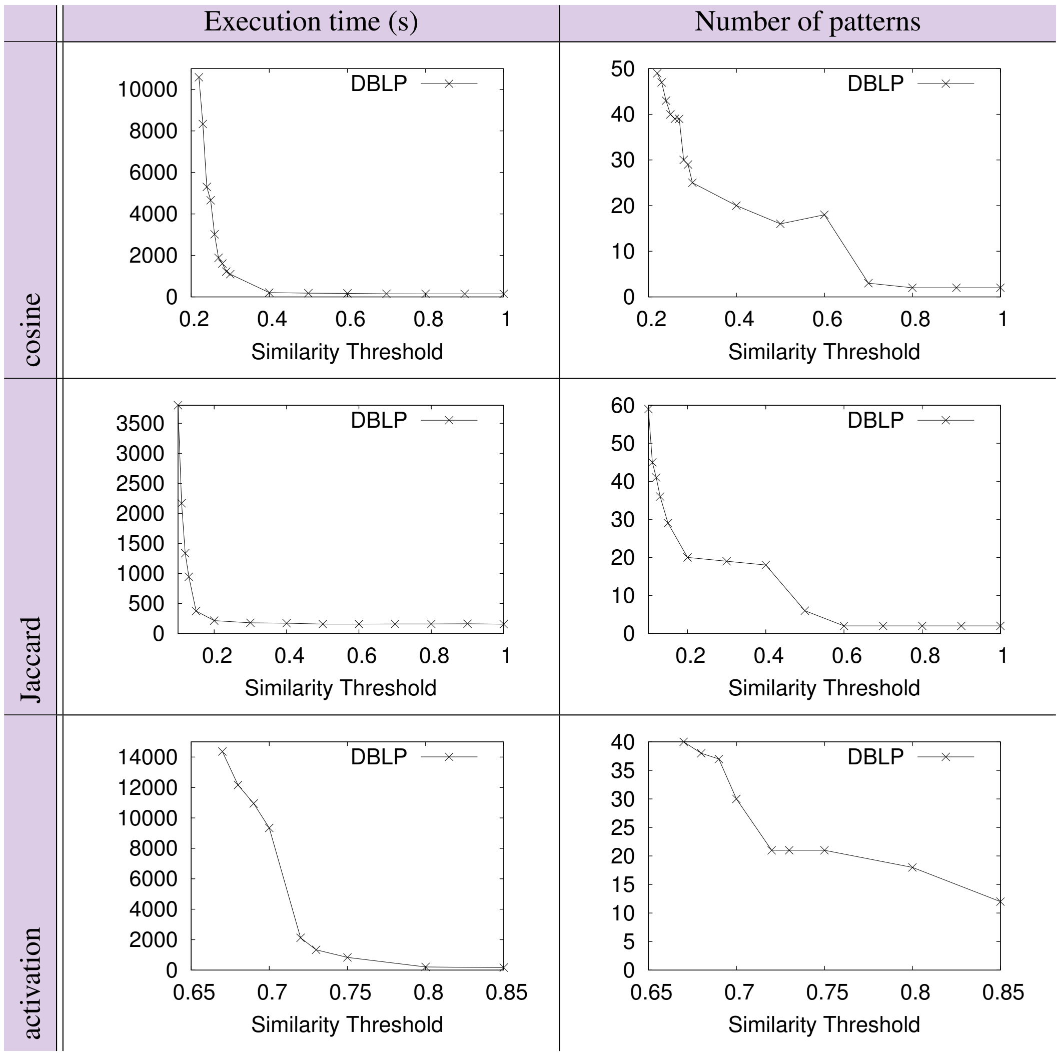


Figure 2.11: Execution times and number of pattern extracted while varying the similarity threshold ($\vartheta = 20$ for cosine and Jaccard $\vartheta = 30$ for activation and signature).

Qualitative experiments

Here are presented some experiments done on the "DBLP" dataset to illustrate the method.

First, we wanted to compare the close neighbourhood of the authors. We did an experiment with Jaccard similarity and $\sigma = 0.1$ to extract patterns whose vertices have at least 10% of common neighbourhood and $\vartheta = 30$. Figure 2.12 reports the two patterns having the most important number of times in the set of extracted patterns, i.e., those depicting a trend on a longer time.

The first co-evolution pattern (fig. 2.12, left) depicts a set of nine authors close in the co-authorship graph. Their respective number of publications in VLDB conference series decreases whereas their number of publications in PVLDB increases between 2002-2006 and 2004-2008 and between 2004-2008 and 2006-2010. For the sake of simplicity, let us say the number of publications increases/decreases between 2002 and 2010. This pattern reflects the new policy of the VLDB endowment. Indeed, PVLDB appeared in 2008, the review process of the VLDB conference series is done in collaboration with in 2010 and entirely through PVLDB in 2011. It is noteworthy that this pattern is a clique on the two timestamps, indeed all the vertices are co-authors of a paper on databases in 2005, paper which is considered on the two timestamps.

The second pattern (fig. 2.12, right) describes a group of 4 authors which are major actors in the data mining community. They have an increasing number of publications in top data mining/databases conferences between 1998 and 2006. This pattern is also a clique, as these authors are often working together.

Some experimentations were also made using the activation and the cosine similarities, however the patterns extracted are similar to those extracted with the Jaccard similarity.

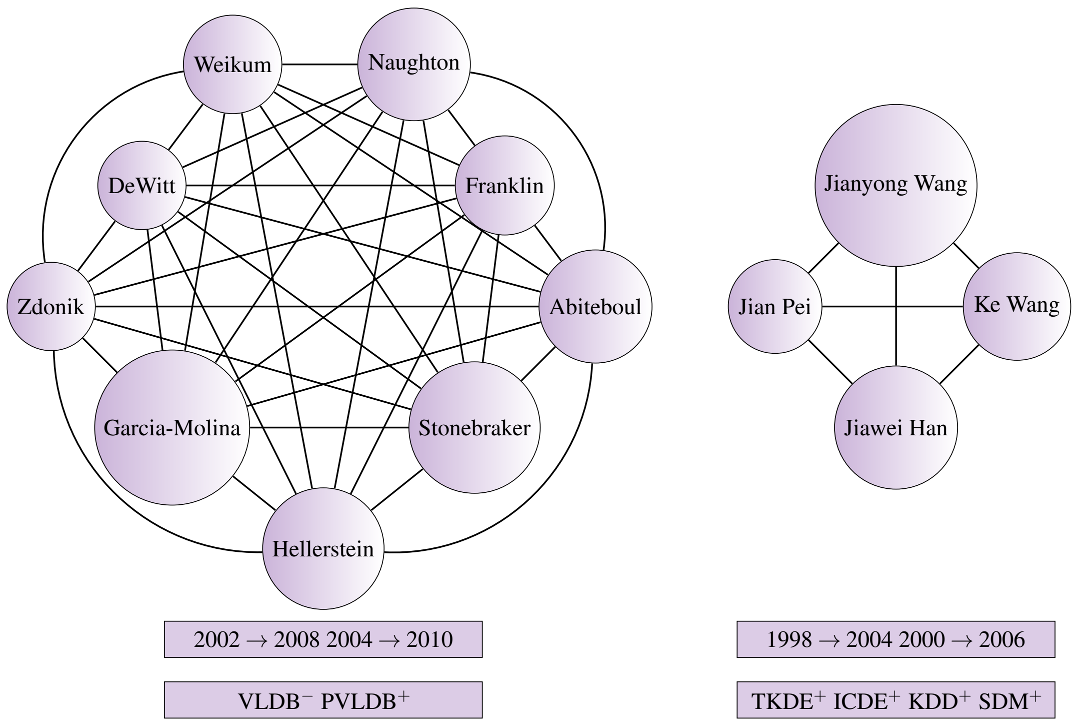


Figure 2.12: Patterns extracted from DBLP with the Jaccard similarity ($\sigma = 0.1$, $\vartheta = 30$)

Second, we wanted to use the signature similarity to compare the structure of the neighbourhood of the authors. Figure 2.13 presents a pattern extracted using the signature similarity with a similarity threshold $\sigma = 0.999$ and a volume threshold $\vartheta = 30$. However, as explained earlier, the algorithm does not scale on this dataset using the signature similarity, this pattern is the first pattern extracted even if the extraction did not end. This pattern identifies rising authors in the bioinformatics area (understood as publishing in journals Bioinformatics and BMC Bioinformatics). Indeed, they have none or few publications in the two journals during the period 2000-2004 and more publications within the period 2002-2006. These authors are not connected, thus this pattern can not be discovered considering Jaccard, cosine or activation measures. It is noticeable that they have no co-authors in the first years, then they have a similar neighbourhood structure. This can be explained easily by the dataset as $bioinformatics$ and $BMCbioinformatics$ are borderland conferences compared to other ones which are mostly oriented on databases and knowledge discovery. Then there are authors in these journals who never published in the other ones and are not related to other communities, or that do not appear in the dataset as they did not publish 10 papers in the set of conferences and journals.

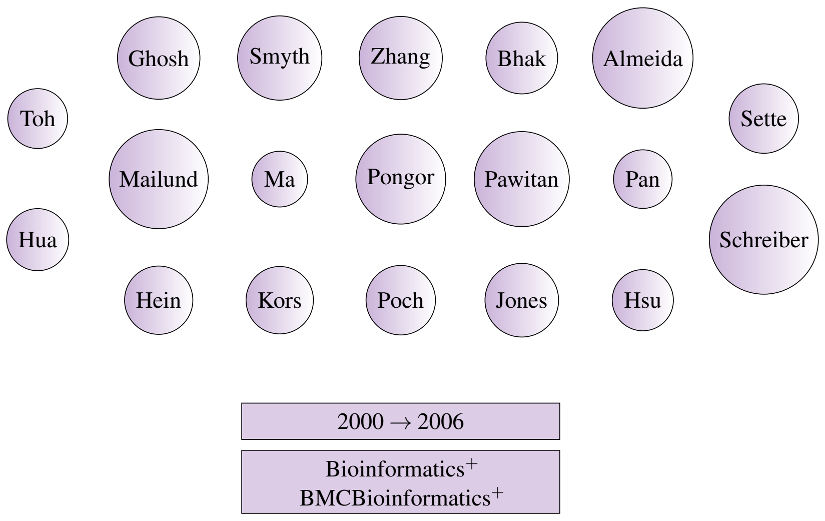


Figure 2.13: Pattern extracted from DBLP with the signature similarity ($\sigma = 0.999$, $\vartheta = 30$)

With this dataset, no valuable experiments were made considering the amount of change in attribute values δ as all the evolution are equals, i.e., authors publish a limited number of papers in a given conference/journal at a given timestamp. However experiments were made on other datasets and will be presented in part III.

2.4.2 Co-evolution Patterns With A Maximum Diameter

Let us now report on experimental results on the "DBLP" dataset using connectivity. Here, the only parameters to set are the size constraints and the diameter constraint. The diameter constraint depends of the needs of the user. Indeed if the user wants a highly connected group of vertices, for instance if he wants a group of persons who know each others, Δ is set to 1 or 2. However if the user is analysing a picture represented as a graph of pixels, and if he wants to find an object even if it's large, Δ is set to $|\mathcal{V}| - 1$ and then connected components are extracted.

Quantitative experiments

We conducted experiments to evaluate the performance of the algorithm depending on the volume threshold ϑ and the diameter threshold Δ. Figure 2.14 reports the results of these experiments.

Whatever Δ, the constraint of volume allows a noticeable reduction of the number of patterns and of the execution times. However, the volume threshold has to be fixed considering the value of Δ. Indeed, a pattern constrained to have a diameter equal to 1, i.e. to be a clique, is much smaller in terms of number of vertices than a pattern constrained to have only connected vertices. With the "DBLP" dataset, you can find small groups of authors that have all written a paper together, while many authors are connected if they are studying a same domain of research. However the more there are vertices the less they have common trends, which limits the size of the patterns with higher Δ.
The diameter constraint has also an effective impact on the execution times. A small Δ allows a great pruning of the vertices, and times. The greater is Δ, the less effective is the pruning. Moreover, the computation of strict neighbourhood is easy but the computation of larger neighbourhood and the computation of possible pruning is complex and time consuming. That explains the difference in execution times between $\Delta = 1$ and $\Delta = 2$. However, the execution times with $\Delta = |V| - 1$ is faster. Indeed, the simple connectivity allows to not enumerate all the vertices but one. Once $C.T$ and $C.\Omega$ are empty, and considering that $C.V$ has been pruned, then vertices

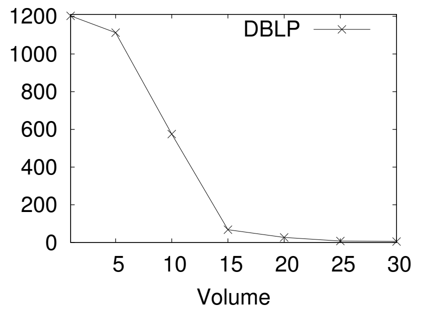
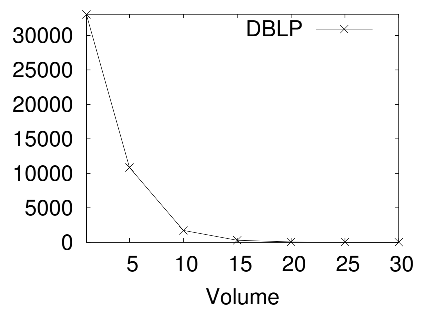
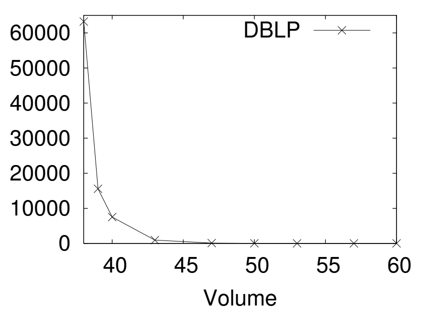
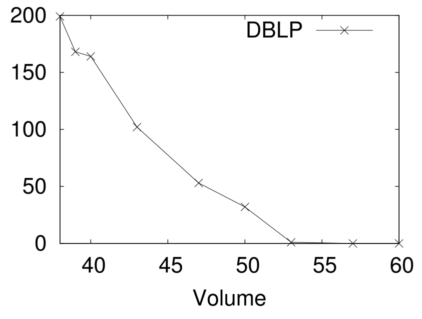
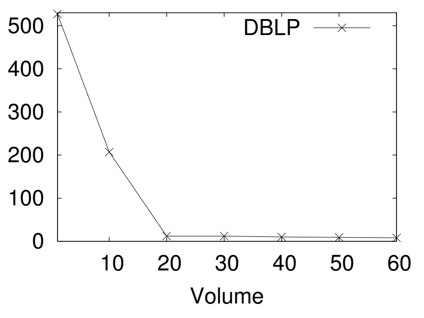
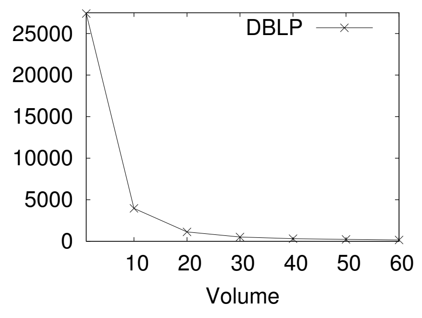
	Execution time (s)	Number of patterns
$\Delta = 1$ (clique)		
$\Delta = 2$		
$\Delta = \|V\|$ (connected comp.)		

Figure 2.14: Number of patterns and execution times on "DBLP" dataset while varying volume, using different Δ ($min_V = 1$,$min_T = 1$ and $min_A = 1$).

of $C.V$ can all be added to P as they respect all the constraints even connectivity while added all together. Even if this reduces the possibility of pruning with the *evol* constraint, it allows a drastic reduction of the execution times. Indeed as the execution with $\Delta = 2$ and $\vartheta = 40$ lasts around 2 hours, the execution with $\Delta = 2144$ and $\vartheta = 40$ lasts only few seconds.

Qualitative experiments

Here are presented some patterns extracted on the "DBLP" dataset to illustrate the method.

First, we wanted to extract groups of authors that are closely related, i.e., that have at least one co-author in common. Figure 2.15 reports four patterns extracted with parameters $\Delta = 2$, $min_V = 4$, $min_T = 2$, $min_A = 2$ and $\vartheta = 30$.

The two first patterns (fig. 2.15, left) are the ones having the most important number of times, i.e., depicting a trend on the longest time. The first pattern (i.e., the 8 authors on the left) and the second one (i.e., the 5 authors on the right) have 4 authors in common which are represented with a darker color. As representing all co-authorship relations would be hardly readable, here are aggregated all connections on the three timestamps. Doing this, the first pattern is nearly a clique, as only "Garofalakis" is not connected to "Stonebraker", "Weikum" and "Garcia-Molina", and the second is a clique plus the author “Cormode” which is only connected to Garofalakis. This reflects that these authors work closely and with same groups of authors. Their respective number of publications in VLDB conference series decreases whereas their number of publications in PVLDB increases during 3 time steps, i.e., between 2002 and 2012. As explained previously, these patterns reflect the new policy of the VLDB endowment, in a longer time period than for the previous patterns.

Notice that the first pattern (i.e., the 8 authors on the left) is similar to the first one found with similarity (see Figure 2.12, p. 61, left), with 6 authors in common, 2 timestamps in common and the same trends.

The two other patterns (fig. 2.15, right) are the ones having the most important number of attributes, i.e., that follow the most similar publication policy. The two sets of authors (i.e., the 4 on the left and the 4 on the right) have 3 authors in common represented with a darker color. They are all major actors in the data mining community, and are closely related as the two patterns are cliques. The trend shows that their number of publication has increased in top data mining/databases conferences between 1998 and 2006.

Notice that the first pattern (i.e., the 4 authors on the left) is exactly the same than the second one found with similarity (see Figure 2.12, p. 61, right).

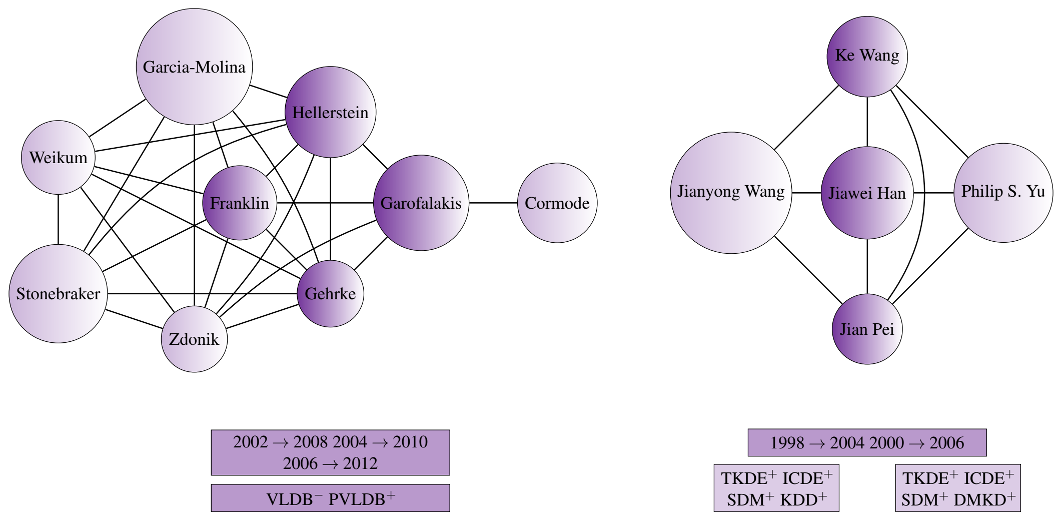


Figure 2.15: Patterns extracted from DBLP with the diameter constraint ($\Delta = 2$, $min_V = 4$, $min_T = 2$, $min_A = 2$ and $\vartheta = 30$)

In a second time we wanted to extract patterns that would depict a more general trend, i.e., which apply to a larger number of authors. We did an experiment with $\Delta = 2144$ to only constrain the pattern to be connected and $\text{min}_V = 215$ to constrain the pattern to contain at least 10% of the authors. Eight patterns were extracted, four of them present trends on “VLDB” or “PVLDB” and can be explained as the preceding patterns, the other four are presented in table 2.1. The two first pattern concern the “CIKM” conference, with an increasing number of publications between 2004 and 2012. This depicts the increasing attractiveness of this conference. The two last patterns concern the “ICDE” conference, with an increasing number of publications between 2002 and 2008 and a decreasing number between 2006 and 2012. This illustrates the difficulty to constantly publish in top conferences.

This dataset is a restrained set of the “DBLP” scientific publication database selected through a set of conferences in databases, data mining and machine learning mainly. Thus using a high Δ leads easily to patterns with a lot of vertices where the authors included in have less

V	T	Ω	
Authors	Time steps	$-$	$+$
273 authors	2004-2010		CIKM
316 authors	2006-2012		CIKM
258 authors	2002-2008		ICDE
268 authors	2006-2012	ICDE	

Table 2.1: Patterns extracted from DBLP with $\Delta = 2145$, $min_V = 215$, $min_T = 1$, $min_A = 1$ and $\vartheta = 1$

importance than the attributes. With this dataset, a small diameter threshold is then more interesting. However, with other dataset, using a higher Δ or even $\Delta = |V| - 1$ can be meaningfull. See part III for more examples.

2.4.3 Discussion On The Results

The two structural relations we defined convey different semantics. Jaccard, cosine or activation similarities provide an information on the amount of common neighbourhood. A small diameter allows to be sure that there is a short path to access from one vertex to the others. These two types of relational constraints make it possible to extract similar types of patterns. For instance, in the previous experiments an identical pattern has been found using Jaccard and the diameter with $\Delta = 2$: $\{$(Jianyong Wang, Jian Pei, Jiawei Han, Ke Wang),(1998 $\rightarrow$ 2004,2000 $\rightarrow$ 2006),(TKDE$^+$,ICDE$^+$,SDM$^+$,KDD$^+$)$\}$. However, the *diameter* constraint is easier to parametrize, indeed, the semantics conveyed by this constraint is more easily interpretable. A small diameter implies a highly connected group of vertices (e.g., a group of persons who know each others), and a larger diameter describes sparser groups until only connected components that can describe large groups of vertices (e.g., an object in a picture). The similarity threshold must be fixed to extract only the patterns with the best similarity, i.e., it is fixed considering the distribution of similarities and not the needs of the user. Indeed, the execution times increase exponentially once a given threshold is achieved. Using the diameter, Δ can be fixed considering the type of pattern wanted. The most meaningful thresholds are often small ones ($\Delta = 1$ or $\Delta = 2$) to extract cliques or quasi-cliques or the highest one ($\Delta = |\mathscr{V}| - 1$) to extract connected components. Considering the execution times, the higher the Δ the higher the execution times (except for $\Delta = |\mathscr{V}| - 1$).

The signature similarity allows to extract pattern that have a similar structure in their neighbourhood even if they have no common neighbour. It can be used to extract vertices with a certain role, for instance, new authors or “hubs” connecting communities. This cannot be extracted using diameter, whatever the threshold. However the similarity values are really high, then it does not allow an efficient pruning of the search space and is in practice not usable.

Using a large diameter allows to extract vertices that are connected but not closely. In social datasets, as a co-authorship graph, or even the internet [58], the diameter is known to be small. Then a mean Δ can allow to extract communities and a large Δ can allow to extract behavior of a large part of the dataset. With other datasets, as spatio-temporal datasets, it can be really meaningful. For instance, with vertices as segments of a picture which are connected to their neighbours, it can enables to extract an object as a conjunction of its segments. Then the use of the diameter with $\Delta = |V| - 1$ enables to extract patterns that can not be found using similarity. Indeed, the similarity is not transitive, then it is difficult to find large set of entities that are similar in pairs.

2.5 Conclusion

As presented in this chapter, co-evolution patterns can be extracted in an acceptable execution time and can provide relevant information with both proposed constraints on the graph structure. However, as presented in section 2.4, the diameter constraint is easier to parametrize than the similarity constraint. Then even if similarity allows to extract patterns conveying a different semantics, the diameter will be used in the rest of the manuscript as *related* constraint.

We observe that the collection of extracted patterns may be large, may contain similar patterns (e.g., the 4 patterns concerning "VLDB" in the last extraction) and sometimes patterns providing obvious information considering the rest of the graph (e.g., the 9 authors changing from "VLDB" to "PVLDB" in the first extraction). We address these limits in the next chapter. We propose interestingness measures on the pattern to extract only the most relevant ones considering the dataset. We propose the extraction of hierarchical co-evolution patterns, i.e., patterns that provide a more general information and avoid some redundancy in the collection of patterns. The introduction of new constraints lead to an increasing of the number of parameters to set and the difficulty to fix too many thresholds leads us to propose the extraction of sky-patterns [96] according to a subset of measures (i.e., user preferences).

3 — Toward More Relevant Patterns

The chapter is organized as follows. The first section defines density constraints to assess the specificity of the patterns with respect to the three dimensions (vertices, times and attributes) and the impact of these measures is evaluated through an empirical study. In section 3.2, we propose to exploit a hierarchy on the attributes to extract hierarchical co-evolution patterns. We define the hierarchy, hierarchical co-evolution patterns and new constraints, we propose an algorithm to extract the new patterns and we evaluate it in an empirical study. Finally, in section 3.3, we propose to use skyline analysis to extract the best patterns considering their values over the different measures. This allows both to consider only the patterns that maximize user-preferences and to avoid thresholding problems.

3.1 Relevant Patterns With Regard To The Dynamic Attributed Graph

In this section we introduce new interestingness measures to guide the search toward more relevant patterns. Such measures have been studied in itemset mining. For instance Morishita and Sese [64] define a theoretical framework to compute significant association rules according to statistical measures and Kuznetsov [55] defines the stability of a formal concept. In [37], the authors propose a survey of interestingness measures in data mining. However they have not been studied in our knowledge in dynamic attributed graphs.

Given a dynamic attributed graph $\mathscr{G} = (\mathscr{V}, \mathscr{T}, \mathscr{A})$, the idea is to discard patterns that depict a behaviour which is not specific and that is supported by many other elements of the graph. Figure 3.1 illustrates a pattern as a cube in the dataset and highlights the outside elements we are considering in each measure. Given a co-evolution pattern $P = (V, T, \Omega)$, 3 measures of "outside densities" are proposed to evaluate the behaviour of outside vertices, outside times and outside attributes on the elements of the pattern. The measures we propose aim at answering the following questions:

- **Vertex specificity:** How similar are the vertices outside the trend dynamic sub-graph to the ones inside it?
- **Temporal dynamic:** What about the dynamic of the pattern? Does it appear suddenly?
- **Trend relevancy:** Do the vertices of the pattern co-evolve only on the attributes of the pattern?

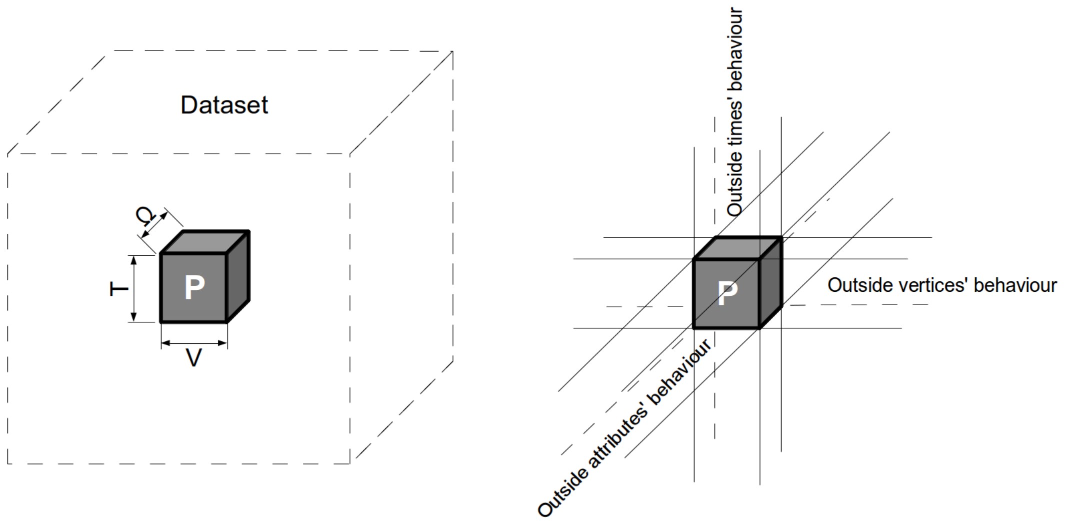


Figure 3.1: Outside densities overview

3.1.1 Interestingness Measures

Let us now present more in details the measures. The measure of vertex specificity compares the behaviour of the outside vertices $\mathscr{V} \setminus V$ to the trends of Ω at times of T. The measure of temporal dynamic checks if the behaviour of vertices of V for attributes of Ω is really specific to the timestamps of T. And the measure of trend relevancy computes how similar is the behaviour of vertices of V at timestamps of T for outside attributes $\mathscr{A} \setminus A$.

Vertex specificity:

With this measure, the aim is to quantify the average proportion of trends that are satisfied by the vertices that do not belong to the pattern. Let us consider Figure 3.1 as a representation of the pattern in the dataset by a a three-dimension matrix where there is a 1 if $trend(v,t,a^s)$ is respected and 0 otherwise. We want to study the outside vertex behaviours by evaluating the proportion of 1 among all the values. The intuition behind this measure can be easily depicted by a social network where timestamps are years and attributes are characteristics of the persons, a pattern that would describe a set of friends whose age is increasing is not relevant, indeed the age of all the persons of the graph is increasing. To illustrate with the “DBLP” dataset, the aim is to evaluate the proportion of authors that have the same publication policy for the conferences and years of the pattern.

Definition 3.1.1 — Vertex Specificity (κ). Given $\delta_{condition}$ the Kronecker function that is equal to 1 if *condition* is satisfied, or 0 otherwise, $P = (V,T,\Omega)$ a pattern and $\kappa \in [0,1]$ a user defined threshold:

$$\begin{aligned} vertexSpecificity(P) &= \frac{\sum_{v \in \mathscr{V} \setminus V} \sum_{t \in T} \sum_{a^s \in \Omega} \delta_{a^s(v,t)}}{|\mathscr{V} \setminus V| \times |T| \times |\Omega|} \\ specificityV(P,\kappa) &= true \Leftrightarrow vertexSpecificity(P) \leq \kappa \end{aligned} \tag{3.1}$$

The more the behaviour depicted by the timestamps and the attribute trends of the pattern is specific to the vertices of the pattern, the lower this measure. For instance, with the “DBLP” dataset, lots of patterns are extracted whose authors have a decreasing number of publications in the VLDB conference and an increasing number of publications in PVLDB between 2002 and 2010. This is due to the new policy of the “VLDB endowment”. Such patterns will have a relatively high value of vertex specificity. But a pattern that describes an increasing number of publications in a conference that increased its selection criterion will have a small vertex specificity value.

■ **Example 3.1** Given $\mathscr{G}$ the graph in figure 2.2 (p.44), and $\kappa = 0.25$, Figure 3.2 represents the elements considered to compute vertex specificity. The coloured vertices are the outside vertices, the dark coloured attributes are those that respect the trends and the light coloured attributes are those that do not. Indeed we want to depict the vertices that respect the trends of the pattern.

- Given $P = \{(v_1, v_2, v_3), (t_2), (a_2^-, a_3^-)\}$, $vertexSpecificity(P) = \frac{3}{4} = 0.75$, then $specificityV(P, \kappa)$ is false, P is not a valid pattern.
- Given $Q = \{(v_2, v_4), (t_1), (a_1^-, a_3^+)\}$, $vertexSpecificity(Q) = \frac{1}{6} = 0.17$, then $specificityV(Q, \kappa)$ is true, Q is a valid pattern.

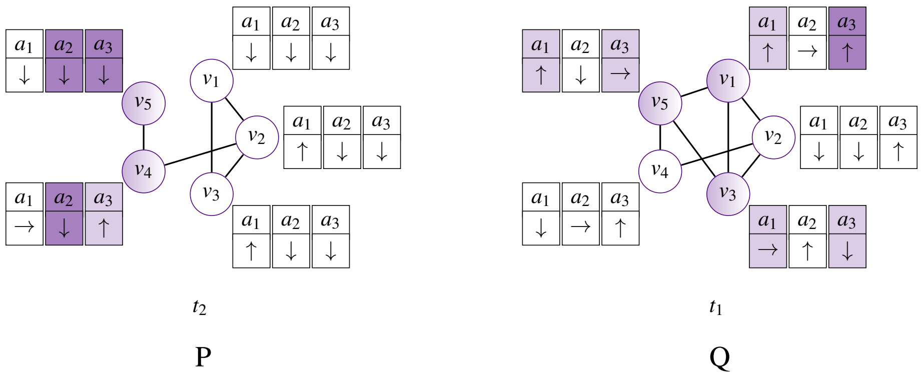


Figure 3.2: Illustration of Example 3.1

■

Temporal dynamic

The aim of this is to evaluate the dynamic of the proportion of vertices and attributes that satisfy the pattern before, between and after the timestamps of T. Considering Figure 3.1, we want to study the outside times’ behaviour. However, instead of vertex specificity, we do not want to evaluate the global proportion but how the behaviour appear in time, i.e., we want to consider the outside time where the number of “1” compared to the number of possible “outside vertices triset” is the highest. If we look back to the example with “age” as an attribute, it is obvious than the age do not increase only at these timestamps. To illustrate with the “DBLP” dataset, the aim is to take notice if there is at least one other timestamp where the authors follow these trends on the attributes.

Definition 3.1.2 — Temporal Dynamic (τ). Given $\delta_{condition}$ the Kronecker function, $P = (V, T, \Omega)$ a pattern and $\tau \in [0, 1]$ a user defined threshold:

$$temporalDynamic(P) = \max_{t \in \mathscr{T} \setminus T} \frac{\sum_{v \in V} \sum_{a^s \in \Omega} \delta_{a^s(v,t)}}{|V|.|\Omega|} \tag{3.2}$$
$$dynamicT(P, \tau) = true \Leftrightarrow temporalDynamic(P) \leq \tau$$

The more the co-evolution pattern bursts, the lower this measure. For instance, with the “DBLP” dataset, if the trends of the pattern have ever been followed by 80% of the authors at the previous timestamp, the value of temporal dynamic will be high, indeed the pattern does not burst and this behaviour is not specific to these timestamps. But a pattern that describes a complete change of publication policy of a group of authors will have a small temporal dynamic value.

■ **Example 3.2** Given $\mathcal{G}$ the graph in figure 2.2 (p.44), and $\tau = 0.25$, Figure 3.3 represents the elements considered to compute temporal dynamic. The coloured vertices are the vertices of the pattern at timestamps outside the pattern and as for the vertex specificity example, the dark coloured attributes are the one that respect the trends and the light coloured attributes are the ones that do not. We want to depict how the trends are respected at timestamps outside the pattern.

- Given $P = \{(v_1, v_2, v_3), (t_2), (a_2^-, a_3^-)\}$, $temporalDynamic(P, \tau) = \frac{2}{6} = 0.33$, then, $dynamicT(P)$ is false, P is not a valid pattern.
- Given $Q = \{(v_2, v_4), (t_1), (a_1^-, a_3^+)\}$, $temporalDynamic(Q, \tau) = \frac{1}{4} = 0.25$, then, $dynamicT(Q)$ is true, Q is a valid pattern.

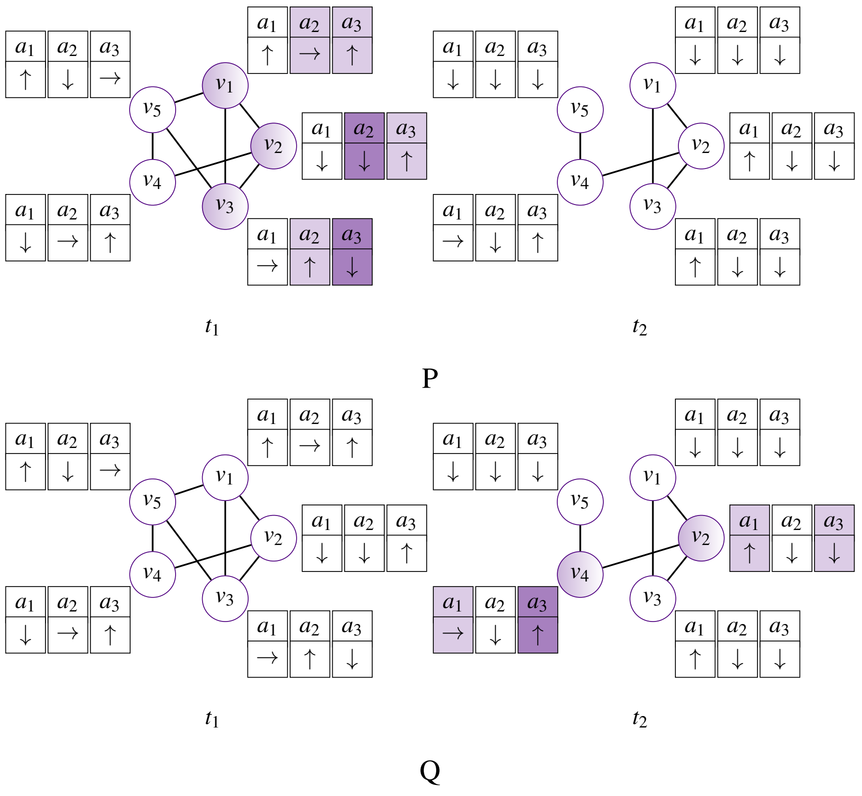


Figure 3.3: Illustration of Example 3.2

■

Trend relevancy:

The aim of this measure is to evaluate the entropy of the outside attribute trends and consider the one that has the smallest entropy. Considering Figure 3.1, we want to study the outside attributes' behaviour. With attributes, we can not compute a real density, indeed, we do not have a trend associated to outside attributes and computing the proportion of "1" would have no sense. But it is interesting to evaluate if the vertices of the pattern follow coherent trends on outside attributes, i.e., if they follow the same trends on the same attributes at the same times. To this aim, we use a measure of entropy, commonly understood as a measure of disorder or a measure of the uncertainty in a random variable. The Shannon entropy of a finite sample is written $H(X) = -\sum_i P(x_i) \times log_b(P(x_i))$ [14]. To illustrate with the "DBLP" dataset, the aim is to evaluate how similar is the publication policy of the authors at these timestamps in other conferences.

Definition 3.1.3 — Trend Relevancy (ρ). Given $\delta_{condition}$ the Kronecker function, $P = (V, T, \Omega)$ a pattern and $\rho \in [0,1]$ a user defined threshold:

$$\begin{aligned} E(a^s, V, T) &= \frac{\sum_{v \in V} \sum_{t \in T} \delta_{a^s(v,t)}}{\sum_{v \in V} \sum_{t \in T} \left(\delta_{a^-(v,t)} + \delta_{a^+(v,t)}\right)} \\ trendRelevancy(P) &= \min_{a \in \mathscr{A} \setminus A} \sum_{s \in \{-,+\}} -E(a^s, V, T) \log E(a^s, V, T) \\ relevancyT(P, \rho) &= true \Leftrightarrow trendRelevancy(P) \geq \rho \end{aligned} \tag{3.3}$$

The more a co-evolution pattern is trend relevant, the higher this measure. For instance, with the "DBLP" dataset, if the pattern concerns two authors who publish nearly always together, the value of rend relevancy will be low as they have a common publication policy in almost all conferences. But a pattern, that describes authors who collaborate on a work but write papers in parallel in different conferences, may have a high trend relevancy value.

■ **Example 3.3** Given $\mathscr{G}$ the graph in figure 2.2 (p.44), and $\rho = 0.25$, Figure 3.4 represents the elements considered to compute trend relevancy. The coloured vertices are the vertices of the pattern at timestamps of the pattern, instead of other examples, the dark coloured attributes are the increasing trends and the light coloured attributes are the decreasing trends, while constant trends are not coloured. We want to depict the difference of behaviour with regard to outside attributes.

- Given $P = \{V, T, \Omega\} = \{(v_1, v_2, v_3), (t_2), (a_2^-, a_3^-)\}$,

$$\begin{aligned} trendRelevancy(P) &= \min_{a \in \{a_1\}} \sum_{s \in \{-,+\}} -E(a^s, (V), (T)) \times log((E(a^s, (V), (T)) \\ &= -\frac{2}{3} \times log(\frac{2}{3}) - \frac{1}{3} \times log(\frac{1}{3}) \\ &= 0.28 \end{aligned}$$

  then $relevancyT(P, \rho)$ is true, P is a valid pattern.
- Given $Q = \{V', T', \Omega'\} = \{(v_1, v_2, v_4), (t_1), (a_3^+)\}$,

$$\begin{aligned} trendRelevancy(Q) &= \min_{a \in \{a_1, a_2\}} \sum_{s \in \{-,+\}} -E(a^s, (V'), (T')) \times log(E(a^s, (V'), (T')) \\ &= min\left(-\frac{2}{3} \times log(\frac{2}{3}) - \frac{1}{3} \times log(\frac{1}{3}), -\frac{1}{3} \times log(\frac{1}{3}) - \frac{0}{3} \times log(\frac{0}{3})\right) \\ &= min(0.28, 0.16) = 0.16 \end{aligned}$$

  then $relevancyT(Q, \rho)$ is false, Q is not a valid pattern.

■

3.1.2 MINTAG Algorithm

The algorithm MINTAG follows the same principles as Algorithm 1 (p. 53). It integrates constraints related to the interestingness measures we defined. These constraints are neither monotonic nor anti-monotonic considering the algorithm, however some properties of the constraints can be used to prune the search space.

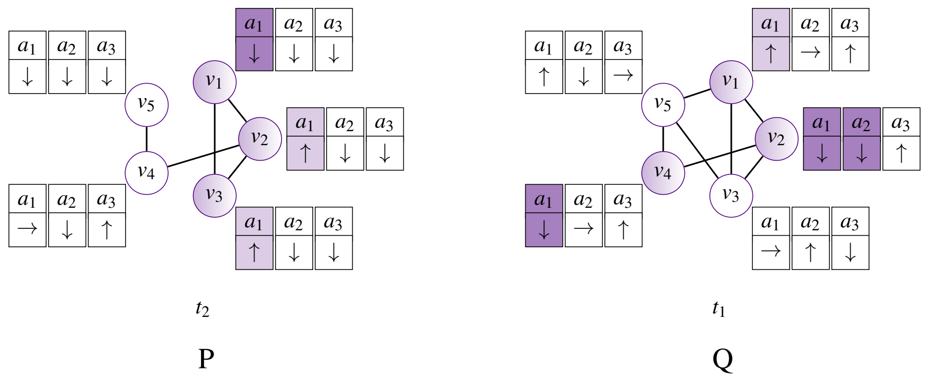


Figure 3.4: Illustration of Example 3.3

specificityV:

This constraint is monotonic or anti-monotonic on each of its parameters. Considering the equation $\frac{\sum_{v\in\mathscr{V}\setminus V_1}\sum_{t\in T_1}\sum_{a^s\in\Omega_1}\delta_{a^s(v,t)}}{|\mathscr{V}\setminus V_2|\times|T_2|\times|\Omega_2|} \leq \kappa$, it is monotonic on V_1, T_2 and Ω_2 and anti-monotonic on V_2,T_1 and Ω_1.Then we can define an anti-monotonic lower bound $LBvertexSpecificity(P,C)$ which is lower than the vertex specificity of all final patterns P' such that P' is a specialization of P created with P and elements of C:

$$LBvertexSpecificity(P,C) = \frac{\sum_{v\in\mathscr{V}\setminus(P.V\cup C.V)}\sum_{t\in P.T}\sum_{a^s\in P.\Omega}\delta_{a^s(v,t)}}{|\mathscr{V}\setminus P.V|\times|P.T\cup C.T|\times|P.\Omega\cup C.\Omega|}$$
$$LBvertexSpecificity(P,C) \leq vertexSpecificity(P')$$

If $LBvertexSpecificity(P,C)$ is greater than the threshold κ, all possible patterns P' have a vertex specificity value greater than κ. Then, enumeration can be stopped as no valid pattern can be enumerated.

dynamicT:

As for the vertex specificity, this constraint is monotonic or anti-monotonic on each of its parameters. We then propose a lower bound $LBtemporalDynamic(P,C)$ which is anti-monotonic on its parameters on the numerator and monotonic on the ones on the denominator. $LBtemporalDynamic(P,C)$ is lower than the temporal dynamic of all final patterns P' such that P' is a specialization of P created with P and elements of C:

$$LBtemporalDynamic(P) = \max_{t\in\mathscr{T}\setminus(P.T\cup C.T)} \frac{\sum_{v\in P.V}\sum_{a^s\in P.\Omega}\delta_{a^s(v,t)}}{|P.V\cup C.V||P.\Omega\cup C.\Omega|}$$
$$LBtemporalDynamic(P,C) \leq temporalDynamic(P')$$

If $LBtemporalDynamic(P,C)$ is greater than the threshold τ, all possible patterns P' have a temporal dynamic value greater than τ. Then, enumeration can be stopped as no valid pattern can be enumerated.

relevancyT:

Handling this constraint is a little more difficult. Let us first consider the entropy function with two probability values: $f(x) = -x\log(x) - (1-x)\log(1-x)$. This function increases on $[0,\frac{1}{2}]$

and decreases on $[\frac{1}{2}, 1]$.

First, using this notation, $relevancyT$ can be rewritten as $\min_{a \in \mathscr{A} \setminus A} f(E(a^+, V, T)) \geq \rho$ or $\min_{a \in \mathscr{A} \setminus A} f(E(a^-, V, T)) \geq \rho$. Indeed, the function of trend relevancy is defined on two parameters $+$ or $-$.
Second, we can derive the following bounds on $E(a^s, V, T)$:

- $UB(a^s) = \frac{\sum_{v \in P.V \cup C.V} \sum_{t \in P.T \cup C.T} \delta_{a^s(v,t)}}{\sum_{v \in P.V} \sum_{t \in P.T} \left(\delta_{a^-(v,t)} + \delta_{a^+(v,t)}\right)}$
- $LB(a^s) = \frac{\sum_{v \in P.V} \sum_{t \in P.T} \delta_{a^s(v,t)}}{\sum_{v \in P.V \cup C.V} \sum_{t \in P.T \cup C.T} \left(\delta_{a^-(v,t)} + \delta_{a^+(v,t)}\right)}$

$LB(a^s) \leq E(a^s, V, T) \leq UB(a^s)$. UB is monotonic on its numerator parameters, and anti-monotonic on its denominator ones and inversely for LB.

Thus, if $UB(a^s) \leq \frac{1}{2}$, then f is increasing and $f(E(a^s, V, T)) \leq f(UB(a^s))$. Similarly, if $LB(a^s) \geq \frac{1}{2}$, then f is decreasing and $f(E(a^s, V, T)) \leq f(LB(a^s))$. Then these bounds can be used to prune the search space:

Theorem 3.1.1 If $\exists a \in \mathscr{A} \setminus (P.V \cup C.V)$ and $s \in \{+, -\}$ s.t.

$$OR \begin{cases} UB(a^s) \leq \frac{1}{2} & \wedge \quad f(UB(a^s)) < \rho \\ LB(a^s) \geq \frac{1}{2} & \wedge \quad f(LB(a^s)) < \rho \end{cases} \tag{3.4}$$

then, $f(E(a^s, V, T)) < \rho$ and $relevancyT$ is false and we can conclude that no valid pattern can be enumerated.

3.1.3 Empirical Study

Let us now report some experimental results on the "DBLP" dataset using MINTAG algorithm. The proposed quantitative experimental study aims at evaluating the impact of the constraints on the efficiency of the algorithm in terms of execution times and number of patterns. As the use of these constraints reduce the computation times we took the opportunity to experiment on replications of the dataset and study how the algorithm scales when the dataset size increases. A qualitative experiment is also proposed to assess that the addition of these constraints allows to extract interesting patterns.

Quantitative Results

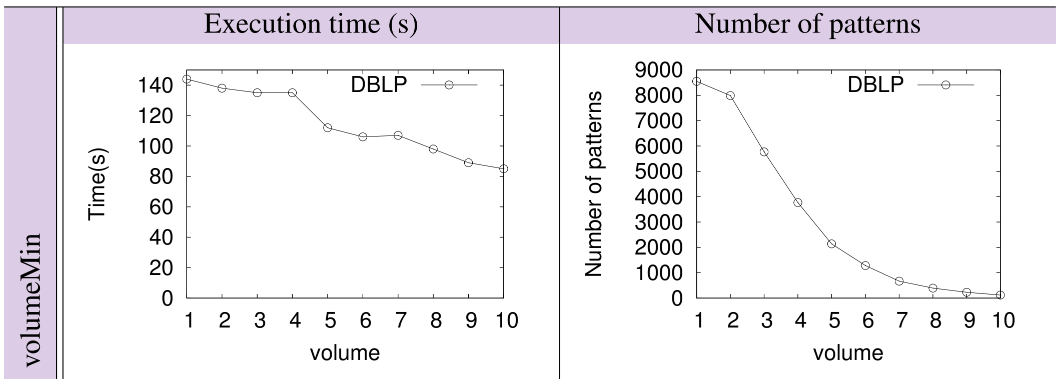


Figure 3.5: Number of patterns and runtime for "DBLP" dataset with respect to volume with parameters $\Delta = 2, \kappa = 0.5, \tau = 0.8$ and $\rho = 0.05$.

First we made an experiment on the volume threshold to compare it to the one in figure 2.14 (p.63) with $\Delta = 2$. Figure 3.5 shows the number of extracted patterns and the execution times of MINTAG on the "DBLP" dataset with respect to the volume threshold. The other parameters were set to $\Delta = 2, \kappa = 0.5, \tau = 0.8$ and $\rho = 0.05$. When the minimum volume threshold decreases, more execution time is required since more co-evolution patterns are obtained. Yet, MINTAG is able to extract co-evolution patterns when the minimum volume threshold is minimal, i.e., when $\vartheta = 1$. When compared to 2.14 (p.63), we can see that both the number of patterns and the execution times are much smaller. Indeed, the execution times and number of patterns with $\vartheta = 40$ and without outside densities is similar to the ones with $\vartheta = 1$ and the use of outside densities. Though, the thresholds are not set in a way to highly constrain the execution as many pattern are still found.

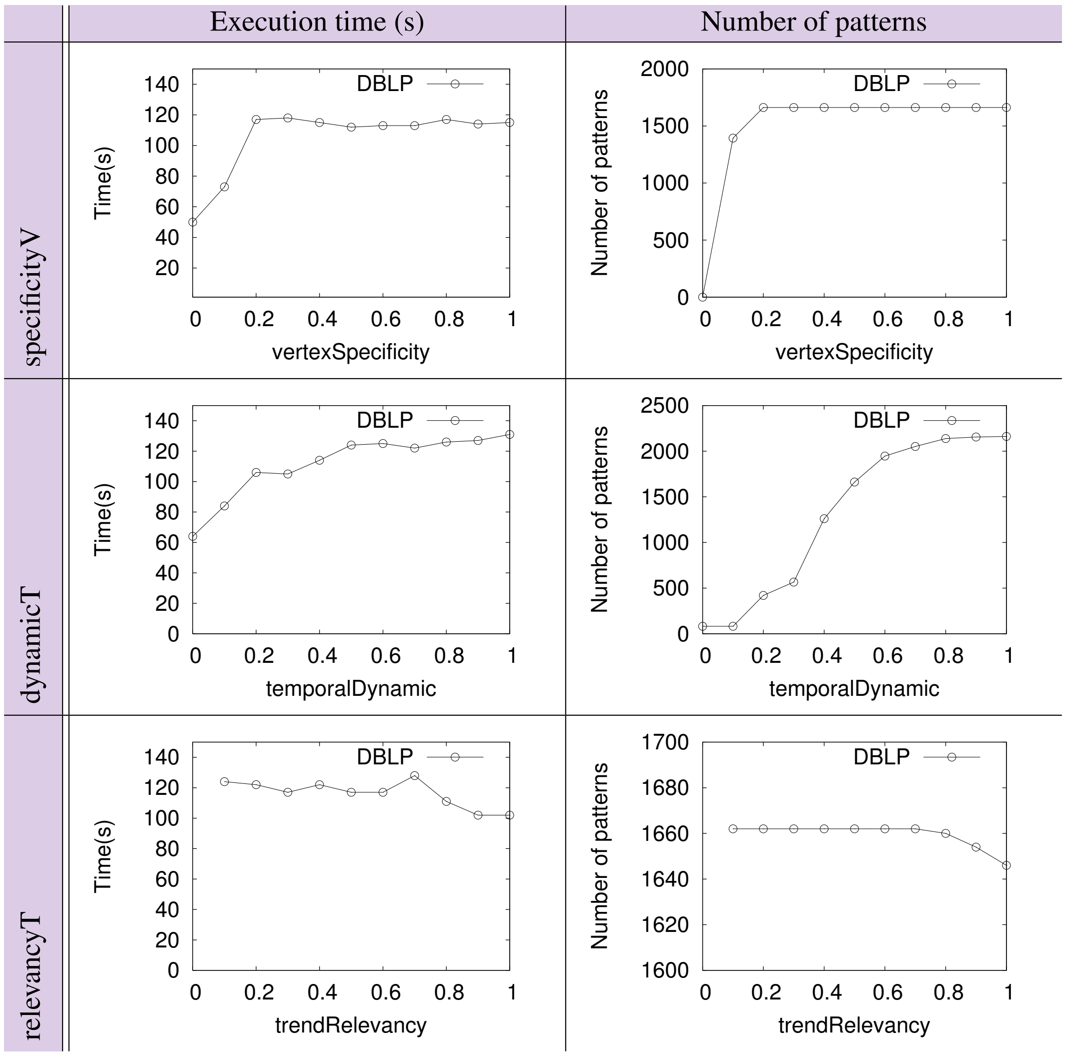


Figure 3.6: Execution time and number of patterns with respect to the specificity measures. When not varying, thresholds were set to: $\kappa = 0.3$, $\tau = 0.5$, $\rho = 0.1$, $\vartheta = 5$ and $\Delta = 2$.

Let us now present the results of experiments while varying the outside densities thresholds. Figure 3.6 reports the execution times and the number of patterns with respect to vertex specificity, trend relevancy and temporal dynamic. We can observe that the less stringent the constraints, the higher the execution times and the number of patterns. With vertex specificity and temporal dynamic, the execution time is divided by two between *threshold* $= 1$ and *threshold* $= 0$, this behavior shows that the approach pushes efficiently these constraints that are neither monotonic nor anti-monotonic. It is noteworthy that the execution time of MINTAG on DBLP with the trend relevancy threshold ρ set to 0 is not available because the process was killed after several hours. The thresholds are set to $\Delta = 2$, $\kappa = 0.3$ and $\tau = 0.5$ which is not much stringent , the execution times are then similar to those of subsection 2.4.2 (p.62) and, as the volume threshold is low, it explodes.

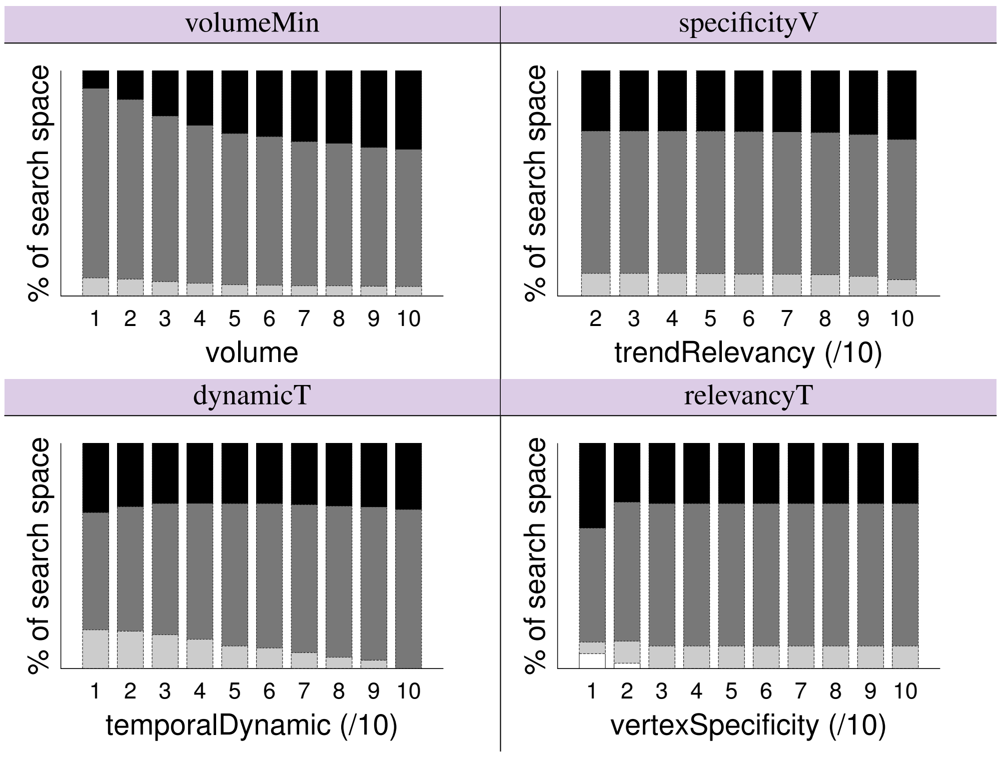


Figure 3.7: Constraint efficiency on the "DBLP" dataset w.r.t. specificity measures. From top to bottom: volume (black), relevancyT (dark grey), dynamicT (light grey) and specificityV (white). The parameters are set to the same threshold than in figures 3.5 and 3.6.

Let us now study the effectiveness of each constraint on the "DBLP" dataset, when varying the different thresholds (volume, vertex specificity, temporal dynamic and trend relevancy). To this end, we count the number of unpromising candidates pruned by each constraint. The results are shown in Figure 3.7. It is noteworthy that all the constraints enable to prune unpromising candidates. The volume constraint makes possible to prune large part of the search space. This behaviour is expected since this constraint is anti-monotonic. We can observe that the trend relevancy constraint is effective, as it prunes almost 50% of the unpromising candidates on DBLP in most of the cases. Even if this constraint has no anti-monotonic property, it is efficiently pushed in MINTAG. However, this large percentage is partly due to the specificities of the dataset which contains many '0' in its attributes. The pruning impact of the *dynamicT* constraint is not negligible, since it prunes nearly 20% of the candidates. The vertex specificity has however a smaller impact on the search space which can be explained by the dataset, indeed, except for major event as the apparition of a new conference, the behavior of the authors is really specific to each one. These experiments are made on the "DBLP" dataset, and results are depending on the dataset, indeed, in paper [29] we proposed an experiment on another dataset, namely "One year around 9/11" which is presented in appendix A and studied in Part III of the manuscript. In the Figure 6. of this paper, experiments on both datasets are presented and we can see that the impact of *volume* and *specificityV* is less important while the impact of *dynamicT* and *relevancyT* is much more important.

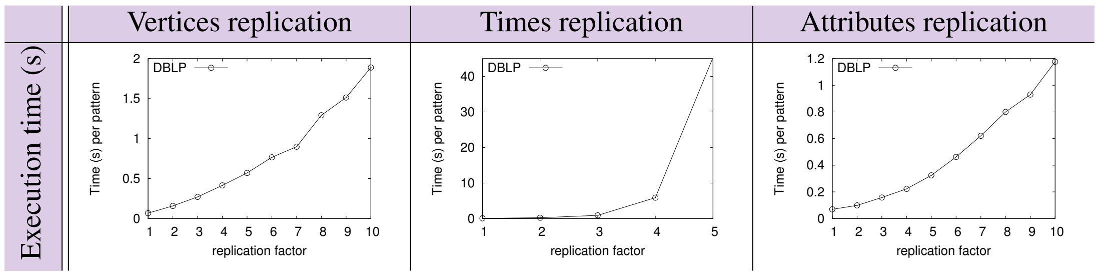


Figure 3.8: Execution times per pattern with respect to replication factors on vertices, attributes and time stamps ($\kappa = 0.3$, $\tau = 0.5$, $\rho = 0.1$, $\vartheta = 5$ and $\Delta = 2$.

Figure 3.8 reports on the scability of MINTAG. We used DBLP and replicated alternatively the number of vertices, timestamps and attributes. As the number of extracted patterns is not preserved by these replications (i.e., the vertex replication adds connected components while the time replication introduces new variations involving the last time stamp) we report the runtime per pattern. It appears that MINTAG is more robust to the increase of the number of attributes and to the number of vertices than to the number of time stamps. This is a good point since, in practice, the number of vertices is often large while the number of attributes and mainly the number of time stamps are rather small.

Qualitative Results

We carry out an extraction taking into account all the specificity constraints : $\Delta = 2$, $\vartheta = 5$, $\kappa = 0.1$, $\tau = 0.3$, $\rho = 0.1$, $min_V = 2$ and $min_T = min_A = 1$. We obtained 11 patterns in 189 seconds. First notice that all these pattern are specific with regard to the graph, indeed the vertex specificity and temporal dynamic thresholds are binding and they all have a trend relevancy greater than 0.7. Let us study the pattern with the highest number of attributes and the pattern with the best specificity measure on each of the 3 measures, they are presented in Figure 3.9.

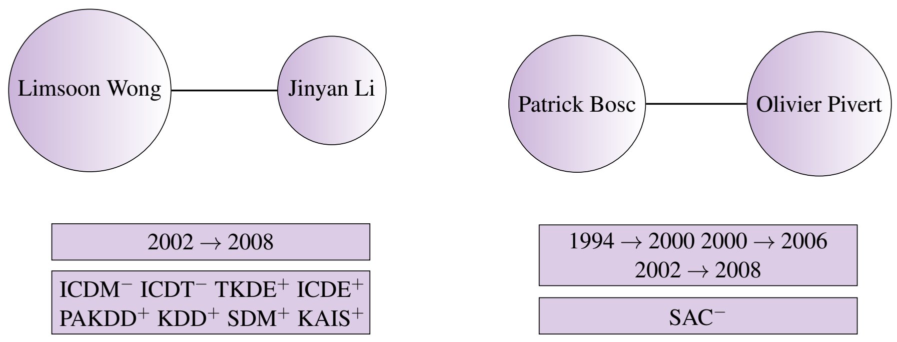


Figure 3.9: Some patterns extracted from DBLP with the parameters: $\Delta = 2$, $\vartheta = 5, \kappa = 0.1$, $\tau = 0.3$, $\rho = 0.1$, $min_V = 2$ and $min_T = min_A = 1$.

The first pattern on the left is the one with the most attributes. It involves 2 authors Limsoon Wong and Jinyan Li who decreased their number of publication in "ICDM" and "ICDT" and increased their number of publications in "TKDE", "ICDE", "PAKDD", "KDD", "SDM", "KAIS" between 2002 and 2008. All these conferences and journals are important ones in the data-mining community and this pattern reflects a publication policy oriented to the best conferences and journals and with success at these years. The second pattern on the right concerns two authors Patrick Bosc and Olivier Pivert that decreased their number of publications in "AAAI" between 1994 and 2000 and then between 2000 and 2008 continuously. This pattern is really specific, $vertexSpecificity = 0.024$, $temporalDynamic = 0$ and $trendRelevancy = 1$ but reflects the difficulty to continuously publish in a same conference.

3.2 Hierarchical Co-Evolution Patterns

The use of user knowledge in the extraction of information can lead to more relevant patterns. The use of hierarchies is studied since a long time. For instance Srikant and Aggrawal [97] propose the extraction of generalized sequential patterns in a database of sequences. In [41], Han and Fu use a taxonomy to extract multi-level association rules. More recently some studies use user-knowledge as hierarchies to study graphs. For instance, Inokuchi [43] proposes to extract generalized frequent sugraphs in labelled graphs using a taxonomy on vertices and edge labels. The authors of [18] define the taxonomy-superimposed graph mining problem, they extract frequent patterns in labelled graphs. None of these methods study either dynamic or attributed graphs. We propose to use a hierarchy on the attributes of the graph to extract hierarchical co-evolution patterns and obtain a smaller collection of patterns where a hierarchical co-evolution pattern can resume several co-evolution patterns.

3.2.1 Hierarchy On The Attributes

A hierarchy $\mathscr{H}$ on $\mathscr{A}$ is a tree where the edges are a relation is_a, i.e., specialization (resp. generalization) corresponds to a path from the root to the leaves (resp. from the leaves to the root) in the tree. The node $\mathscr{A}ll$ is the root of the tree and attributes of $\mathscr{A}$ are the leaves. Different functions are defined to run through the hierarchy:

- $parent(x)$ returns the direct parent of the node x
- $children(x)$ returns the direct children of the node x
- $up(x)$ returns all the ancestors of x, x included, $\mathscr{A}ll$ excluded
- $down(x)$ returns all the descendants of x, x included, $\mathscr{A}ll$ excluded
- $leaf(x)$ returns all the leaves descendants of x, i.e., $down(x) \cap \mathscr{A}$

Moreover the domain of the hierarchy $dom(\mathscr{H})$ is defined as all the nodes except the root, i.e., $dom(\mathscr{H}) = down(\mathscr{A}ll)$. This hierarchy is an a priori knowledge of the expert.

Figure 3.10 presents an example of hierarchy on a toy example. To illustrate it on a real-world dataset, the hierarchy used for the dataset "DBLP" is presented in appendix A.

The root of the hierarchy is not in the domain of the hierarchy. It makes possible to consider unrelated attributes as, for instance, the number of publication of a scientist and its hair color. Unrelated categories of attributes will appear as root children.

The hierarchy is a tree where each node has at most one parent. The use of a forest or a directed acyclic graph could also be studied. However the constraints we define thereafter are not valid on these types of hierarchies, moreover they would increase the computation complexity.

3.2.2 Hierarchical Co-Evolution Patterns

As for co-evolution patterns, intuitively, a hierarchical co-evolution pattern P is a dynamic subgraph of $\mathscr{G}$ induced by (V,T) whose vertices follow the same trend over a subset of attributes. The difference is in the set of attributes which is not only $\mathscr{A}$ but $dom(\mathscr{H})$, i.e., a hierarchical co-evolution pattern is a co-evolution pattern where $A \subseteq dom(\mathscr{H})$. Formally:

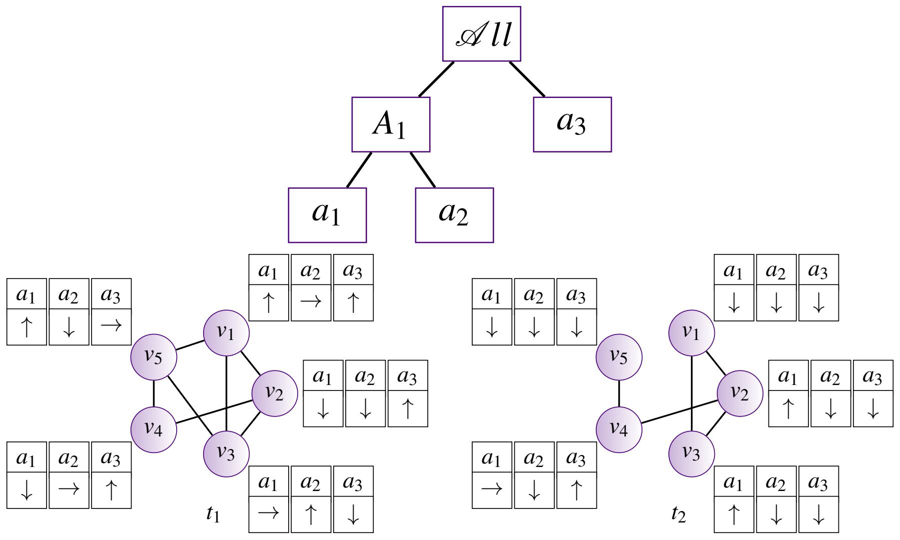


Figure 3.10: Hierarchical toy example.

Definition 3.2.1 — Hierarchical co-evolution Pattern. Given a graph $\mathscr{G} = (\mathscr{V}, \mathscr{T}, \mathscr{A})$ and a hierarchy $\mathscr{H}$, a hierarchical co-evolution pattern is a triplet $P = (V, T, \Omega)$ s.t.:

- $V \subseteq \mathscr{V}$ is a subset of the vertices of the graph.
- $T \subset \mathscr{T}$ is a subset of not necessarily consecutive timestamps.
- Ω is a set of signed attributes, i.e., $\Omega \subseteq A \times S$ with $A \subseteq dom(\mathscr{H})$ and $S = \{+, -\}$ a set of trends.

The two following conditions must hold:

1. Each signed attribute a^s with $a \in A$ and $s \in \{+, -\}$ defines a trend that has to be satisfied by any vertex $v \in V$ at any timestamp $t \in T$: $evolHierarchical(P) = true$,
2. At each time $t \in T$, the vertices of the pattern have to be closely related through the structure E_t of the graph: $related(P) = true$.

The constraint $related(P)$ can be any constraint on the structure of the graph, in this chapter we will use the constraint Δ-$diameter(P)$ (2.2.5, p. 49). The constraint $evolHierarchical(P)$ is an evolution of the constraint $evol(P)$ it can be any constraint on the evolution of the attributes that takes into account the hierarchy. We propose an adaptation of δ-$strictEvol(P)$ (2.2.3, p.47) where the evolution of a "parent" attribute is computed while adding its "children" attribute values. First, let us redefine the trend of a triplet (v,t,a). δ-$trendHierarchical(v,t,a)$ is equivalent to δ-$trend$ (2.2.2, p.47) as it computes the strict difference between values at time t and $t+1$, but a "parent" does not have a value. We choose to consider the value of an attribute at a time t as the sum of the values of its "children" attributes.

Definition 3.2.2 — δ-trendHierarchical. Given a threshold $\delta \in [0,1]$, $G_t(v,a)$ and $G_{t+1}(v,a)$ being two values of attribute a of vertex v at timestamp t and $t+1$. Similar evolution instead of strict evolution is defined as:

$$\begin{cases} \text{iff } (1+\delta) \times \sum\limits_{a_i \in leaf(a)} G_t(v,a_i) < \sum\limits_{a_i \in leaf(a)} G_{t+1}(v,a_i) & \Leftrightarrow \delta\text{-}trendHierarchical(v,t,a) \quad = + \\ \text{iff } (1-\delta) \times \sum\limits_{a_i \in leaf(a)} G_t(v,a_i) > \sum\limits_{a_i \in leaf(a)} G_{t+1}(v,a_i) & \Leftrightarrow \delta\text{-}trendHierarchical(v,t,a) \quad = - \\ \text{else} & \Leftrightarrow \delta\text{-}trendHierarchical(v,t,a) \quad = \varnothing \end{cases} \tag{3.5}$$

Then let us define $strictEvolHierarchical(P)$ which is true if the trends of the attributes of P are respected by all the vertices of the pattern at all the timestamps of P:

Definition 3.2.3 — δ-strictEvolHierarchical(P). Given $P = (V,T,\Omega)$ and the function δ-$trendHierarchical$, δ-strictEvolHierarchical(P) is valid iff:

$$\forall v \in V,\ \forall t \in T \text{ and } \forall a^s \in \Omega \text{ then } \delta\text{-}trendHierarchical(v,t,a) = s \tag{3.6}$$

As for δ-$strictEvol$, we will mainly refer to $strictEvolHierarchical$ which is equivalent to 0-$strictEvolHierarchical$.

This constraint version of $evolHierarchical(P)$ is additive, i.e., a "parent" attribute value is the addition of its "children" attribute values. However it could be defined differently as all hierarchies do not respect this supposition. For instance, the trend associated to a "parent" attribute could be the maximal trend of its "children" attributes, i.e., if a majority of the "children" attributes respect trend $+$, then the "parent" attribute respects trend $+$. Such a proposition has been done in our paper [27]. However, all the real-world datasets we work with have an additive hierarchie, and we claim it is the case of most real-world ones, therefore, we prefer to not develop this version of the constraint here.

The introduction of a hierarchy implies to define new constraints. Indeed, we need to be sure a pattern with a "parent" attribute resumes well the information and we need to choose between a "parent" pattern or a "children" pattern which one is the best. To this aim, we define two measures: purity and gain.

Purity of a pattern

The hierarchy allows to resume the information as the "parent" attribute value is an aggregation of its "children" attribute values. Then, even if the trend of the pattern is true for a "parent" attribute, it is important to check that this is a valid information, i.e., that the trend associated to the "parent" attribute is not influenced by only a small part of its children. Indeed if a "children" attribute has a large increasing in its value and the other "children" attributes have a really small decreasing, the parent attribute will show an increasing but it is not a valuable resume. An example can be seen in figure 3.11. The points are the attribute values at each timestamp and the additive value for B. The trends are represented by the lines between points. We can see that the trends of attribute B are merely induced by the trends of b_1 even if they are different from the ones of the two other "children" attributes b_1 and b_2.

Then, we define $purity(P)$ which makes sure that the pattern is a valid summary, i.e., the trend of the attributes are respected by a large part of their "children" attributes. To this aim, we consider the "children" attributes, we compute the proportion of valid triplet (v,t,a^s), i.e.,

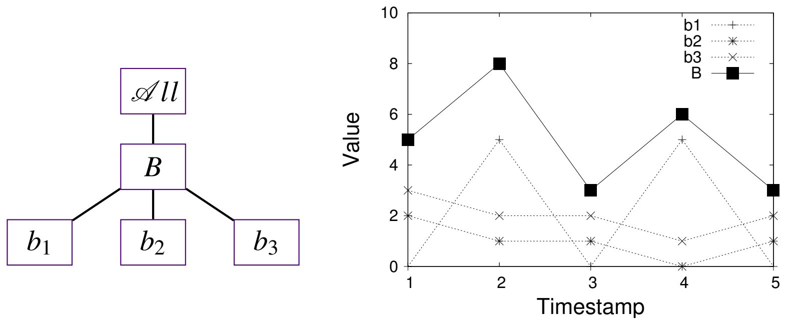


Figure 3.11: Possible impact of a minority of the "children" attribute trends.

such that $trend(v,t,a) = s$ and $a \in leaf(\alpha)$ where $\alpha^s \in \Omega$, with regard to the number of possible triplets.

Definition 3.2.4 — Purity (ψ). Given the Kronecker function $\delta_{condition}$ and given a user-defined threshold $\psi \in [0,1]$, the purity of the pattern is defined as:

$$\begin{aligned} purity(P) &= \frac{\sum_{v \in V} \sum_{t \in T} \sum_{a^s \in leaf(\Omega)} \delta_{a^s(v,t)}}{|V| \times |T| \times |leaf(\Omega)|} \\ purityMin(P,\psi) &\Leftrightarrow purity(P) \geq \psi \end{aligned} \tag{3.7}$$

A pattern that respects *purityMin* is a pattern considered as pure enough to provide a valid and relevant information. For instance with the "DBLP" dataset, a pattern that describes the increasing number of publications in data-mining because the author has changed its publication policy and passed from 0 to several publications in data mining conferences is interesting. However a pattern that describes an increasing number of publications in data mining conferences whereas it increases in only one of them and stayed constant in the other is less interesting than the same pattern with the given conference.

■ **Example 3.4** Given $\mathscr{G}$ and $\mathscr{H}$ the ones in figure 3.10, $\Delta = 5$ and $\psi = 0.6$, Figure 3.12 represents the elements considered to compute purity. The coloured vertices are the vertices of the pattern, the dark coloured attributes are the "children" attributes that respect the trends and the light coloured attributes are those that do not. Indeed we want to depict the proportion of trends respected.

- Given $P = \{(v_1,v_2,v_3,v_4,v_5),(t_2),(A_1^-)\}$, $purity(P) = \frac{7}{10} = 0.7$ then $purityMin(P,\psi)$ is true, P is a valid pattern.
- Given $Q = \{(v_1,v_2,v_3,v_4,v_5),(t_1,t_2),(A_1^-)\}$, $purity(Q) = \frac{11}{20} = 0.55$ then $purityMin(Q,\psi)$ is false, Q is not a valid pattern.

■

Gain of purity of a pattern

Another question to answer is the best level of attributes to keep between the "parent" and the "children" attributes. Indeed, we do not want to extract together a pattern with a "parent" attribute and one or several patterns with a subset of its "children" attributes. Figure 3.13 represents two patterns in two dimensions, i.e., a line is a vertex, a column is an attribute. A cell is colored in black if the trend of the attribute is respected by the vertex, in grey otherwise. Considering the pattern on the left, the trends are mainly respected and purity is equal to 0.81. Specializing the

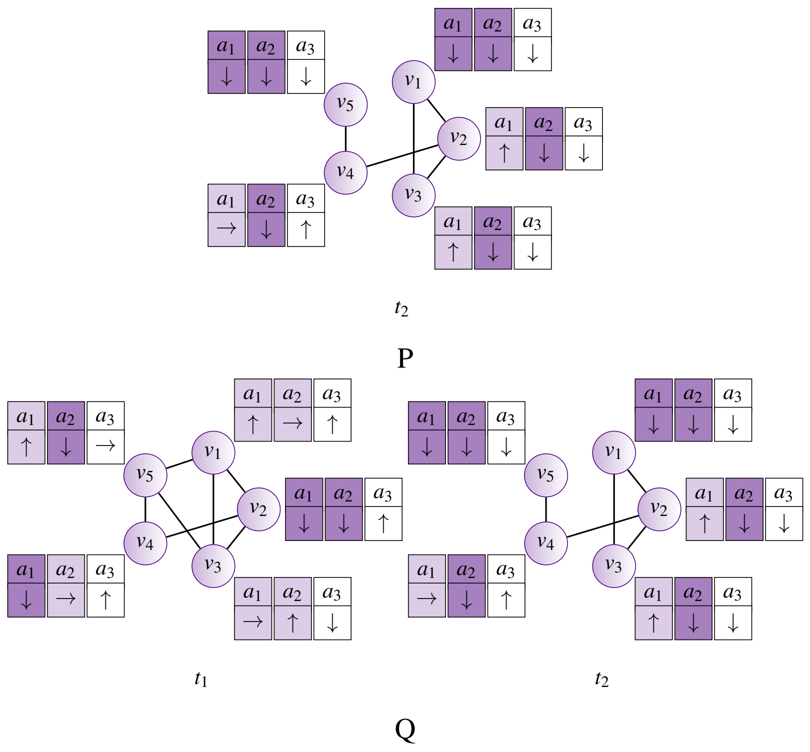


Figure 3.12: Illustration of Example 3.4

attributes would lead to four patterns: $\{(v_2,v_3,v_4)(a_1)\}$, $\{(v_1,v_3,v_4)(a_2)\}$, $\{(v_1,v_2,v_3,v_4)(a_3)\}$ and $\{(v_1,v_2,v_3)(a_4)\}$. Then it seems much more interesting to keep the "parent" attribute A. Considering the pattern on the right, the trends are rarely respected and purity is equal to 0.44. Trends of attributes a_1, a_2 and a_3 are respected by only one vertex, then specializing the attributes would lead to only one interesting pattern: $\{(v_1,v_2,v_3,v_4)(a_3)\}$. This pattern provides an information much more precise than its "parent", then it is here more interesting to specialize the attribute.

The question to answer is then: *Does the gain of purity obtained by specializing the attribute compensates the loss of generality ?*

To this aim we compare the purity of the "children" pattern P with regard to the purity of its "parent" patterns, i.e., all the patterns that contain one "parent" of the attributes of P instead of its "children".

Definition 3.2.5 — Gain of purity (γ). Given a user-threshold $\gamma \geq 1$, the gain of purity is defined as the purity of the "children" pattern compared to the purity of its "parent" patterns:

$$\begin{aligned} gain(P) &\Leftrightarrow \frac{purity(P)}{max_{P_i \in parent(P)}(purity(P_i))} \geq \gamma \\ gainMin(P,\gamma) &\Leftrightarrow gain(P) > \gamma \end{aligned} \tag{3.8}$$

where $P_i \in parent(P)$ if $\exists a_i \in P_i.A$ and $\exists a \in P.A$ s.t. $a \in children(a_i)$ and $(P_i.A \setminus a_i) = (P.A \setminus a)$.

A pattern that respects $gainMin(P)$ is a pattern that provides an information more pure that its "parent" patterns. Let us consider the "children" pattern $Ch = \{(v_1,v_2,v_3,v_4),(t_1),(a_3)\}$ and its only "parent" pattern $Pa = \{(v_1,v_2,v_3,v_4),(t_1),(A)\}$ in Figure 3.13. Considering the figure on

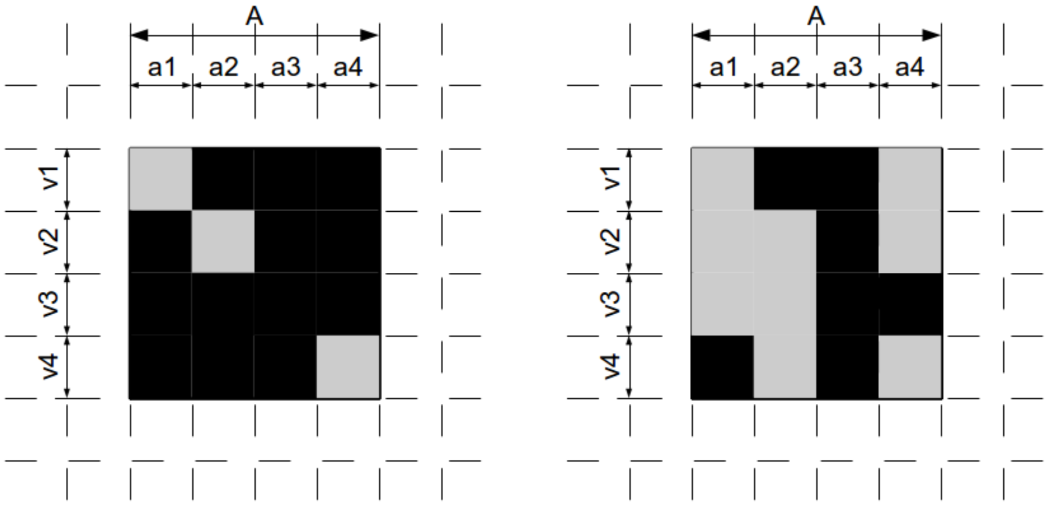


Figure 3.13: Considering a black square is a respected trend and a grey square is not: Do we keep A? Or do we develop a_1 and/or a_2 and/or a_3 and/or a_4?

the left, the gain of purity of Ch is $gain(Ch) = \frac{1}{\frac{13}{16}} = 1.23$, considering the figure on the right, the gain of purity is $gain(Ch) = \frac{1}{\frac{7}{16}} = 2.29$. In the figure on the left the "parent" attribute abstracts well the behaviour of the vertices, then the purity is low compared to the one computed for the figure on the right where the behaviour is mainly followed by attribute a_3.

If no "children" pattern P_i of a pattern P respects $gainMin(P_i)$, then P provides more information than its "children", then the "children" pattern is discarded. It is noticeable that other "descendant" patterns could have a higher purity (purity of a leaf is necessary 1). Yet, the specialization would lead to a large number of possible patterns without necessary more information.

■ **Example 3.5** Given $\mathscr{H}$ the hierarchy in figure 3.10 and $\gamma = 1.2$, Figure 3.14 represents the elements considered to compute purity of the "children" pattern P and the "parent" pattern Q. The coloured vertices are the vertices of the pattern, the dark coloured attributes are the attributes that respect the trends and the light coloured attributes are those that do not. Given $P = \{(v_1, v_2, v_3), (t_2), (a_2^-, a_3^-)\}$, it has only one "parent" pattern $Q = \{(v_1, v_2, v_3), (t_2), (A_1^-, a_3^-)\}$. $purity(P) = 1$ and $purity(Q) = \frac{7}{9} = 0.78$. Then $gain(P) = \frac{1}{0.78} = 1.28$ and $gainMin(P, \gamma)$ is valid. ■

Adaptation Of The Interestingness Measures

The introduction of a hierarchy and of new types of attributes, i.e., attributes that do not have a "real" value in the dataset, implies to adapt the constraints on the pattern. Indeed most of the measures proposed in sections 2.2.3 (p.50) and 3.1.1 (p.68), are not directly compatible with the new definition of the pattern.

Maximality: (see definition 2.2.7, p.50)
The *maximal* constraint is equivalent with or without hierarchy, however, it takes into account any constraint on the pattern, i.e., it also takes into account the new constraints.

Size measures: (see definitions 2.2.8 and 2.2.9, p.51)

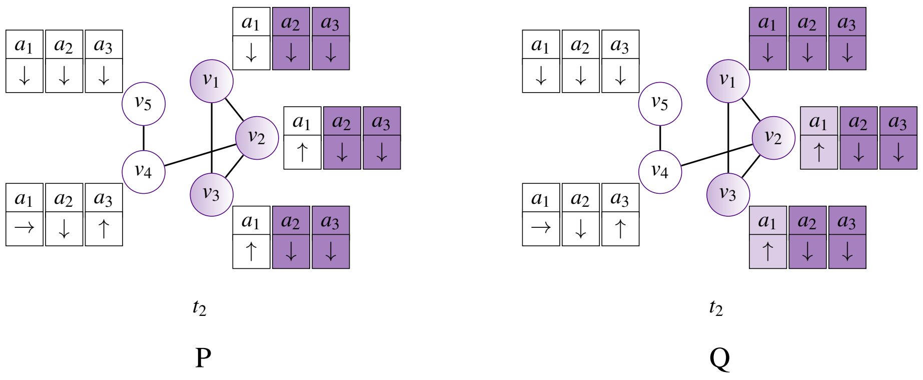


Figure 3.14: Illustration of Example 3.5

The *sizeMinV* and *sizeMinT* constraints are equivalent as there is no change in the definition of the sets of vertices and times. The *sizeMinA* constraint counts the number of attributes in the pattern. However, if the pattern contains only one "parent" attribute but it has many "children" attributes, then the pattern provides most information as one with only 2 of its "children" attributes. Then we redefine the constraint as:

Definition 3.2.6 — *sizeMinA*.

$$sizeMinA(P) \Leftrightarrow sizeA(P) \geq min_A \text{ with } sizeA(P) = |leaf(A)|$$

The volume could also be redefined, but if the trend is respected for the "parent" attribute it is not necessary the case for its "children" attributes. The *sizeMinA* aims to ensure there is enough attributes in the pattern, but the *volume* constraint aims to evaluate the amount of valid triplets, then we do not change the definition.

Specificity measures: (see section 3.1.1, p.68)
Let us now talk about the outside densities constraints, i.e., vertex specificity, temporal dynamic and trend relevancy. The *specificityV* and *dynamicT* constraints are compatible with hierarchical co-evolution patterns, their definition stay the same as 3.1.1 (p.68) and 3.1.2 (p.69). However it is less obvious that *relevancyT* is compatible with hierarchical co-evolution patterns. First, the hierarchy implies a first question: attributes of what level of hierarchy do we consider outside the pattern? Second, some attributes are discarded because they do not provide enough gain of purity, but vertices may follow a same behaviour on part of their children. This contrasts with the concept of trend relevancy that makes sure that the attributes of the pattern are the only ones where vertices of the pattern follow the same trend. Then we do not use trend relevancy for hierarchical co-evolution patterns.

3.2.3 H-MINTAG Algorithm

Algorithm 2 presents the main steps of H-MINTAG. The search space of the algorithm can be represented as a lattice which contains all possible tri-sets from $\mathscr{V} \times \mathscr{T} \times (dom(\mathscr{H}) \times S)$, with bounds $\{\emptyset,\emptyset,\emptyset\}$ and $\{\mathscr{V},\mathscr{T},children(\mathscr{A}ll) \times S\}$. The enumeration can be represented as a tree

Algorithm 2: H-MINTAG

```
Input: $P = \varnothing, C = (\mathscr{V}, \mathscr{T}, children(\mathscr{A}ll) \times S), D = \varnothing, attr, \mathscr{C}$
Output: Hierarchical co-evolution patterns
begin
    if $\neg C.empty$ then
        if $\mathscr{C}(P,C)$ then
            if $\neg(attr = \varnothing)$ then
                $child \leftarrow children(attr)$
                for $i$ in $1..|child|$ do
                    if $gainMin(P.V \cup C.V, P.T \cup C.T, P.A \setminus attr \cup child[i])$ then
                        H-MINTAG($(P \setminus attr) \cup child[i], C \cup child[i+1..|child|], D \cup child[1..i-1]$,
                        $child[i], \mathscr{C}$)
                        $hasSon \leftarrow true$
                if $hasSon$ then
                    for $i$ in $1..|C.A|$ do
                        H-MINTAG($P \cup C.A[i], C \setminus C.A[1..i], D \cup C.A[1..i-1], i, \mathscr{C}$)
                else
                    $attr \leftarrow \varnothing$
            if $attr = \varnothing$ then
                $E \leftarrow ElementTypeToEnumerate(P,C)$
                for $i$ in $1..|C.E|$ do
                    if $E = A$ then
                        $attr \leftarrow C.E[i]$
                    H-MINTAG($P \cup C.E[i], C \setminus C.E[1..i], D \cup C.E[1..i-1], attr, \mathscr{C}$)
                H-MINTAG($P, C \setminus C.E, D \cup C.E, \varnothing, \mathscr{C}$)
    else if $\mathscr{C}(P)$ then
        output $(P)$
```

where each node is a step of the enumeration and a node contains three tri-set P, C and D[1]. At the beginning, $P = D = \emptyset$ and $C = \{\mathscr{V}, \mathscr{T}, children(\mathscr{A}ll) \times S\}$, with $S = \{-,+\}$. The patterns extracted are the one that respect all the constraints presented in the previous sections, i.e., $\mathscr{C} = (coevolution, diameter, sizeMinV, sizeMinT, sizeMinA, volumeMin, maximality, specificityV, dynamicT, purityMin)$ and $gainMin$. The main difference between the two algorithms is the enumeration.

Enumeration

At each step of the enumeration, either an element of C is enumerated (vertex, timestamp or attribute) or an attribute of P is specialized and an attribute from C is enumerated while keeping the non specialized attribute. At the beginning of the algorithm, one vertex, one timestamp and one attribute are enumerated to allow a better use of the constraints to prune the search space. At each step, the elements of C and D (vertices, timestamps and attributes) are deleted if they can not be added to P without invalidating it, i.e., if they can not respect the different constraints. If P does not respect the constraints, the enumeration is stopped.

Figure 3.15 illustrates the enumeration principles, for sake of simplicity the set D is not developed. The specialization step is presented in Figure 3.15 (top), the two children nodes on the left represent the specialization of the attribute, while the two on the right represent the enumeration of a new attribute while keeping the non specialized attribute. During a specialization step of an attribute a^s of P, all the "children" pattern which provide a sufficient

[1]As a reminder: P is the pattern in construction, C contains the elements not yet enumerated and D contains the deleted elements. See Section 2.3 for more details.

gain of purity are enumerated. I.e., $\forall a_i \in children(a)$ s.t. $gain(P_i)$ is respected, the pattern P_i is enumerated and the attributes a_j^s and $a_j^{\bar{s}}$ s.t. $a_j \in children(a)$ with $j > i$ are added to C. All the patterns with attribute a non specialized and an attribute of C are also enumerated, i.e., all the patterns with a and $b \in C.\Omega$. If no children pattern respect $gainMin$ then a node is also enumerated with the "parent" attribute a and no new element.

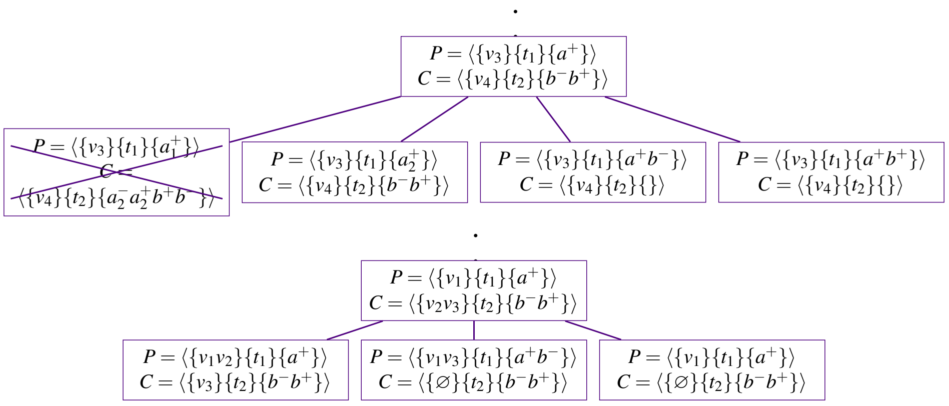

$P = \langle\{v_3\}\{t_1\}\{a^+\}\rangle$
$C = \langle\{v_4\}\{t_2\}\{b^-b^+\}\rangle$
$P = \langle\{v_3\}\{t_1\}\{a_1^+\}\rangle$
$C = \langle\{v_4\}\{t_2\}\{a_2^- a_2^+ b^+ b^-\}\rangle$
$P = \langle\{v_3\}\{t_1\}\{a_2^+\}\rangle$
$C = \langle\{v_4\}\{t_2\}\{b^-b^+\}\rangle$
$P = \langle\{v_3\}\{t_1\}\{a^+b^-\}\rangle$
$C = \langle\{v_4\}\{t_2\}\{\}\rangle$
$P = \langle\{v_3\}\{t_1\}\{a^+b^+\}\rangle$
$C = \langle\{v_4\}\{t_2\}\{\}\rangle$
$P = \langle\{v_1\}\{t_1\}\{a^+\}\rangle$
$C = \langle\{v_2v_3\}\{t_2\}\{b^-b^+\}\rangle$
$P = \langle\{v_1v_2\}\{t_1\}\{a^+\}\rangle$
$C = \langle\{v_3\}\{t_2\}\{b^-b^+\}\rangle$
$P = \langle\{v_1v_3\}\{t_1\}\{a^+b^-\}\rangle$
$C = \langle\{\varnothing\}\{t_2\}\{b^-b^+\}\rangle$
$P = \langle\{v_1\}\{t_1\}\{a^+\}\rangle$
$C = \langle\{\varnothing\}\{t_2\}\{b^-b^+\}\rangle$

Figure 3.15: Example of a specialization step (top) and of an enumeration step (bottom)

The enumeration step is presented in Figure 3.15 (bottom).During an enumeration step, the enumerated element is deleted from C and added to P. If the element is an attribute a^s its symmetric $a^{\bar{s}}$ is also discarded from C.

Properties of the constraints

Only the constraints of size, i.e., $minSizeV$, $minSizeT$ and $minSizeA$ are anti-monotonic, the constraints $coevolution$, $diameter$, $volumeMin$, $specificityV$ and $dynamicT$ are not anti-monotonic considering the enumeration. They can not be used directly to prune the search space. However, the piecewise monotonic property of these constraints can be used to reduce the search space.

coevolution:

The $coevolution$ constraint is not anti-monotonic considering the specialization of an attribute. If a vertex v does not respect the trend of an attribute a^s at time t, it has no meaning on the trend of any "children" attribute a_i. Indeed, the trend associated to a is computed while aggregating the values of the $a_i \in children(a)$, so some a_i can have an opposite trend. However, considering the enumeration of the proposed algorithm, the $coevolution$ can be pruned if the next step is not a specialization step.

If the attributes of the pattern are leaves of $\mathscr{H}$ or if the attributes have already passed the specialization step, the constraint is anti-monotonic. Then enumeration can be stopped if $coevolution(P)$ is false and elements e of C can be deleted if $coevolution(P \cup e)$ is false and $e \notin dom(\mathscr{H}) \setminus \mathscr{A}$.

diameter:

There is no difference for the use of the $diameter$ constraint, so for more details on the pruning of this constraint see section 2.3.2 (p.55).

sizeMin:

The different $sizeMin$ constraints are anti-monotonic. The enumeration can be stopped if $sizeMinV(P \cup C)$, $sizeMinT(P \cup C)$ or $sizeMinA(P \cup C)$ is false.

volumeMin:

The volume constraint is not anti-monotonic considering the enumeration and the specialization, i.e., with $volumeMin(P \cup C)$. Indeed, if the enumeration steps reduce the volume, while specializing an attribute a of P, all the children of a are added to C then $volume(P \cup C)$ increases. We propose an upper bound $UBvolume(P \cup C)$ which is anti-monotonic and higher than the volume of all final patterns P' such that P' is a specialization of P created with P and elements of C:

$$UBvolume(P \cup C) = (P.V + C.V) \times (P.T + C.T) \times leaf(P.\Omega + C.\Omega)$$
$$UBvolume(P \cup C) \geq volume(P')$$

If $UBvolume(P \cup C)$ is lower than the threshold ϑ, all possible patterns P' have a volume value lower than ϑ. Then the enumeration can be stopped as no valid pattern can be enumerated.

purityMin:

The purity constraint is not anti-monotonic. Indeed, while specializing an attribute, the number of valid trisets $(v,t,a^s), v \in (P.V \cup Q.V), t \in (P.T \cup Q.T), a^s \in (P.\Omega \cup Q.\Omega)$ can increase or decrease as the "children" attributes do not all respect the same trend as their parent. One must compute the number of triset that validate either the s or the $\overline{s}$ trend for at least one of the leaf attribute of a. Then the number of valid trisets $\sum_{v \in P.V \cup C.V} \sum_{t \in P.T \cup C.T} \sum_{a \in leaf(P.\Omega \cup C.\Omega)} \sum_{s \in S} \delta_{a^s(v,t)}$ is anti-monotonic and the number of possible trisets $|P.V| \times |P.T| \times |leaf(P.\Omega)|$ is monotonic. We propose an upper bound $UBpurity(P,C)$ which is anti-monotonic and higher than the volume of all final patterns P' such that P' is a specialization of P created with P and elements of C:

$$UBpurity(P,C) = \frac{\sum_{v \in P.V \cup C.V} \sum_{t \in P.T \cup C.T} \sum_{a \in leaf(P.\Omega \cup C.\Omega)} \sum_{s \in S} \delta_{a^s(v,t)}}{|P.V| \times |P.T| \times |leaf(P.\Omega)|}$$
$$UBpurity(P,C) \geq purity(P')$$

If $UBpurity(P \cup C)$ is lower than the threshold ψ, all possible patterns P' have a purity value lower than ψ. Then the enumeration can be stopped as no valid pattern can be enumerated.

specificityV:

If the constraint is equivalent with or without the use of a hierarchy, the lower bound of vertex specificity must be adapted to the use of a hierarchy. Given P' all the final patterns such that P' is a specialization of P created with P and elements of C:

$$LBvertexSpecificity(P,C) = \frac{\sum_{v \in \mathscr{V} \setminus (P.V \cup C.V)} \sum_{t \in P.T} \sum_{a^s \in P.\Omega} \delta_{a^s(v,t)}}{|\mathscr{V} \setminus P.V| \times |(P.T \cup C.T)| \times |leaf(P.\Omega \cup C.\Omega)|}$$
$$LBvertexSpecificity(P,C) \leq vertexSpecificity(P')$$

If $LBvertexSpecificity(P,C)$ is greater than the threshold κ, all possible patterns P' have a purity value greater than κ. Then the enumeration can be stopped as no valid pattern can be enumerated.

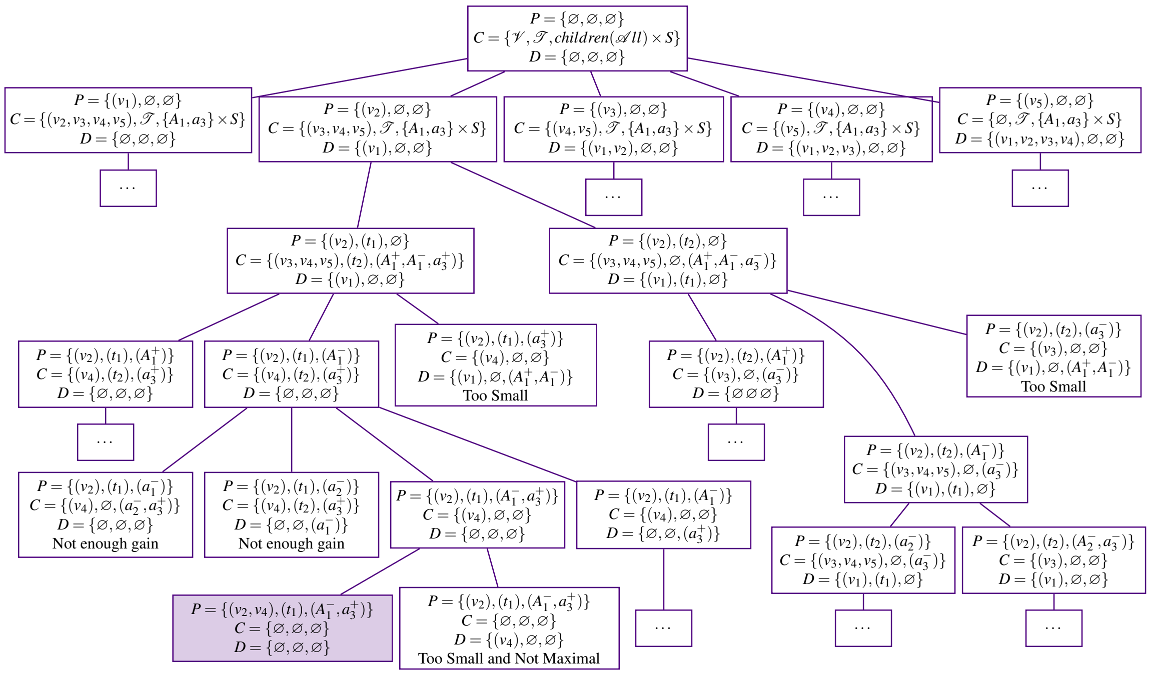
dynamicT:

As for the vertex specificity, the lower bound of temporal dynamic must be adapted to the use of a hierarchy. Given P' all the final patterns such that P' is a specialization of P created with P and elements of C:

$$LBtemporalDynamic(P,C) = max_{t \in (\mathscr{T} \setminus (P.T \cup C.T))} \left(\frac{\sum_{v \in P.V} \sum_{a^s \in P.\Omega} \delta_{a^s(v,t)}}{|P.V \cup C.V| \times |leaf(P.\Omega \cup C.\Omega)|} \right)$$
$$LBtemporalDynamic(P,C) \leq temporalDynamic(P')$$

If $LBtemporalDynamic(P,C)$ is greater than the threshold τ, all possible patterns P' have a purity value greater than τ. Then the enumeration can be stopped as no valid pattern can be enumerated.

3.2.4 Example Of An Execution Trace On A Toy Example



$P=\{\varnothing,\varnothing,\varnothing\}$ $C=\{\mathscr{V},\mathscr{T},children(\mathscr{A}ll)\times S\}$ $D=\{\varnothing,\varnothing,\varnothing\}$
$P=\{(v_1),\varnothing,\varnothing\}$ $C=\{(v_2,v_3,v_4,v_5),\mathscr{T},\{A_1,a_3\}\times S\}$ $D=\{\varnothing,\varnothing,\varnothing\}$
$P=\{(v_2),\varnothing,\varnothing\}$ $C=\{(v_3,v_4,v_5),\mathscr{T},\{A_1,a_3\}\times S\}$ $D=\{(v_1),\varnothing,\varnothing\}$
$P=\{(v_3),\varnothing,\varnothing\}$ $C=\{(v_4,v_5),\mathscr{T},\{A_1,a_3\}\times S\}$ $D=\{(v_1,v_2),\varnothing,\varnothing\}$
$P=\{(v_4),\varnothing,\varnothing\}$ $C=\{(v_5),\mathscr{T},\{A_1,a_3\}\times S\}$ $D=\{(v_1,v_2,v_3),\varnothing,\varnothing\}$
$P=\{(v_5),\varnothing,\varnothing\}$ $C=\{\varnothing,\mathscr{T},\{A_1,a_3\}\times S\}$ $D=\{(v_1,v_2,v_3,v_4),\varnothing,\varnothing\}$
…
…
…
…
$P=\{(v_2),(t_1),\varnothing\}$ $C=\{(v_3,v_4,v_5),(t_2),(A_1^+,A_1^-,a_3^+)\}$ $D=\{(v_1),\varnothing,\varnothing\}$
$P=\{(v_2),(t_2),\varnothing\}$ $C=\{(v_3,v_4,v_5),\varnothing,(A_1^+,A_1^-,a_3^-)\}$ $D=\{(v_1),(t_1),\varnothing\}$
$P=\{(v_2),(t_1),(A_1^+)\}$ $C=\{(v_4),(t_2),(a_3^+)\}$ $D=\{\varnothing,\varnothing,\varnothing\}$
$P=\{(v_2),(t_1),(A_1^-)\}$ $C=\{(v_4),(t_2),(a_3^+)\}$ $D=\{\varnothing,\varnothing,\varnothing\}$
$P=\{(v_2),(t_1),(a_3^+)\}$ $C=\{(v_4),\varnothing,\varnothing\}$ $D=\{(v_1),\varnothing,(A_1^+,A_1^-)\}$ Too Small
$P=\{(v_2),(t_2),(A_1^+)\}$ $C=\{(v_3),\varnothing,(a_3^-)\}$ $D=\{\varnothing\varnothing\varnothing\}$
$P=\{(v_2),(t_2),(a_3^-)\}$ $C=\{(v_3),\varnothing,\varnothing\}$ $D=\{(v_1),\varnothing,(A_1^+,A_1^-)\}$ Too Small
…
…
$P=\{(v_2),(t_2),(A_1^-)\}$ $C=\{(v_3,v_4,v_5),\varnothing,(a_3^-)\}$ $D=\{(v_1),(t_1),\varnothing\}$
$P=\{(v_2),(t_1),(a_1^-)\}$ $C=\{(v_4),\varnothing,(a_2^-,a_3^+)\}$ $D=\{\varnothing,\varnothing,\varnothing\}$ Not enough gain
$P=\{(v_2),(t_1),(a_2^-)\}$ $C=\{(v_4),(t_2),(a_3^+)\}$ $D=\{\varnothing,\varnothing,(a_1^-)\}$ Not enough gain
$P=\{(v_2),(t_1),(A_1^-,a_3^+)\}$ $C=\{(v_4),\varnothing,\varnothing\}$ $D=\{\varnothing,\varnothing,\varnothing\}$
$P=\{(v_2),(t_1),(A_1^-)\}$ $C=\{(v_4),\varnothing,\varnothing\}$ $D=\{\varnothing,\varnothing,(a_3^+)\}$
$P=\{(v_2),(t_2),(a_2^-)\}$ $C=\{(v_3,v_4,v_5),\varnothing,(a_3^-)\}$ $D=\{(v_1),(t_1),\varnothing\}$
$P=\{(v_2),(t_2),(A_2^-,a_3^-)\}$ $C=\{(v_3),\varnothing,\varnothing\}$ $D=\{(v_1),\varnothing,\varnothing\}$
$P=\{(v_2,v_4),(t_1),(A_1^-,a_3^+)\}$ $C=\{\varnothing,\varnothing,\varnothing\}$ $D=\{\varnothing,\varnothing,\varnothing\}$
$P=\{(v_2),(t_1),(A_1^-,a_3^+)\}$ $C=\{\varnothing,\varnothing,\varnothing\}$ $D=\{(v_4),\varnothing,\varnothing\}$ Too Small and Not Maximal
…
…
…

Figure 3.16: Part of an execution of H-MINTAG.

An example of an execution of the algorithm H-MINTAG is proposed in figure 3.16. The graph used is the toy example presented in figure 3.10 (p. 78). The diameter threshold is set to $\Delta = 2$, volume threshold is set to $\vartheta = 4$, purity threshold is set to $\psi = 0.5$ and gain threshold is set to $\gamma = 2$. Other thresholds are set in such a way they do not constraint the patterns, i.e., $min_V = min_T = min_A = 1$ and $\kappa = \tau = 1$. To keep the figure readable, only a part of the enumeration is presented, the other parts are represented by dots.

Some nodes are noted "Too Small" or "Not Maximal" because of volume, size and maximality constraints. More explanation on this notation are presented in section 3.2.5 (p.87). We are interested here in the nodes denoted with "Not enough gain". For instance the node at the last line but one on the left does not provide enough gain of purity. The purity of $P = \{(v_2),(t_1),(a_1^-)\}$ is equal to 1 and the purity of its parent $\{(v_2),(t_1),(A^-)\}$ is equal to $\frac{2}{3}$, then, the $gain(P) = \frac{1}{\frac{2}{3}} = 1.5$ is lower than γ. Finally the coloured node is a valid co-evolution pattern as all the constraints are respected.

3.2.5 Empirical Study

Some experiments were made on "DBLP". The aim of these experiments is to see the impact of the hierarchy and of the constraints on the execution of the algorithm and to see if the extracted patterns are relevant.

Quantitative experiments

To evaluate the impact of each constraint, let us study the execution time and the number of patterns extracted while varying each threshold independently. When fixed, the parameters were

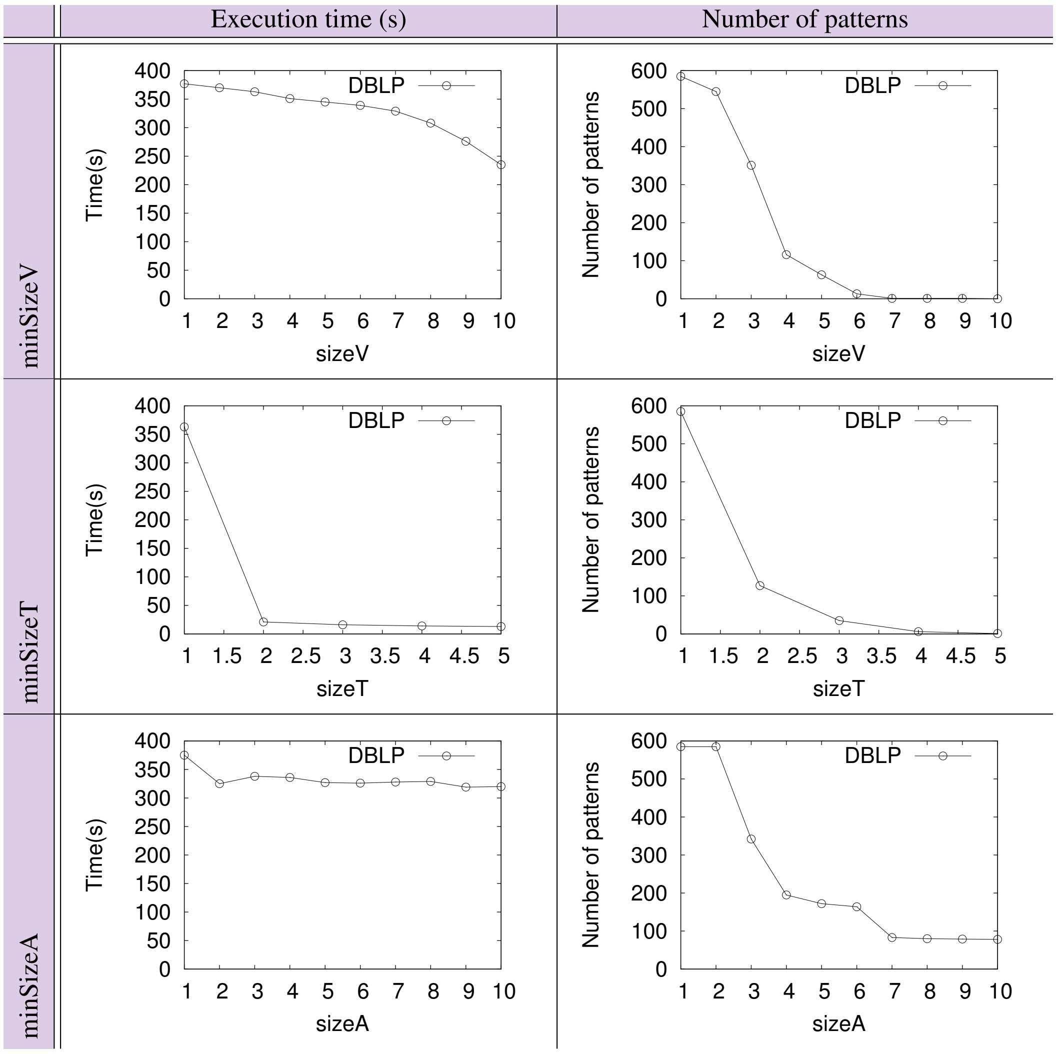


Figure 3.17: Variation of min_V, min_T and min_A.

set to: $\Delta = 1$, $\min_V = min_T = min_A = 1$, $\vartheta = 1$, $\psi = 0.2$, $\gamma = 1.1$, $\kappa = \tau = 0.5$, i.e., in a way to not constrain too much the patterns.

Figure 3.17 introduces the results while varying the size thresholds. All these constraints reduce quickly the number of patterns extracted, however the impact on the running times depends on the constraint. The constraint on the size of the set of vertices allows a reduction of around $\frac{1}{4}$ of the execution times. The constraint on the size of the set of times allows a great reduction between 1 and 2 as the majority of dataset trends are one-off events. The constraint on the size of the set of attributes, however, does not allow a significant reduction of the running times.

Figure 3.18 presents the results while varying the constraint of volume and of outside densities. These 3 constraints also allow a great reduction of the number of patterns extracted but the impact on the running times is limited. The volume constraint allows a really small reduction of the execution times, the vertex specificity does not allow any reduction and the temporal dynamic only allows a reduction when $\tau = 0$, i.e., when only 4 patterns are found. The upper and lower bounds used to prune the search space thanks to these measures are not stringent enough to have

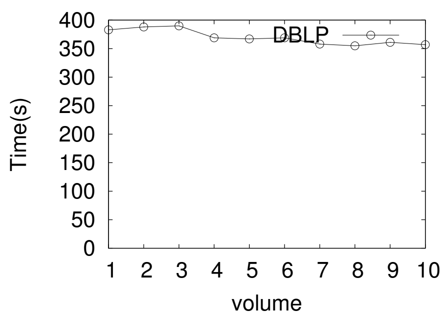
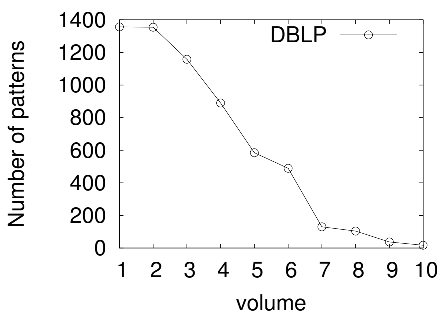
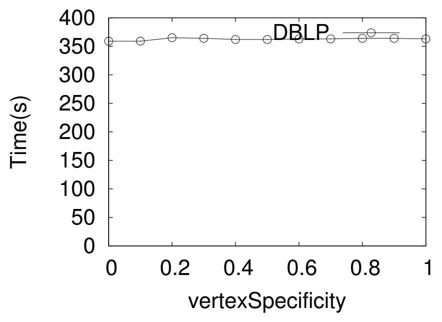
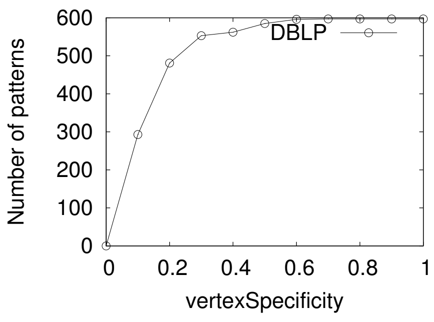
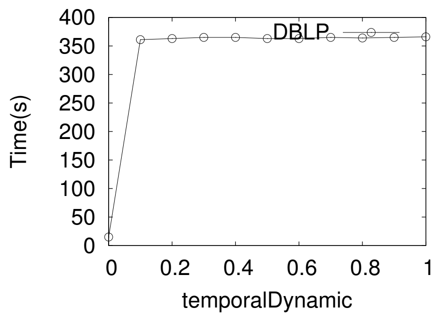
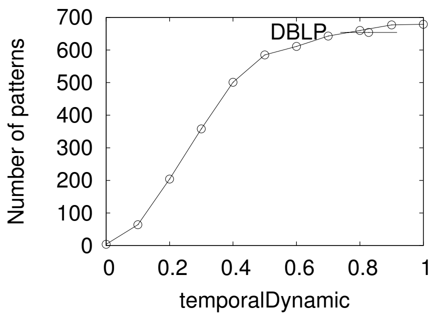
	Execution time (s)	Number of patterns
volumeMin		
specificityV		
dynamicT		

Figure 3.18: Variation of ϑ, κ and τ.

a significant effect on the execution times and only allow to cut branches deep in the tree.

The impact of the hierarchy can be analysed while varying the thresholds of purity and of gain and the depth of the hierarchy. Results are presented in figure 3.19. Purity and gain allow a large reduction of the number of patterns. The purity constraint allows a significant reduction, we can also notice that the curve of the execution times has a similar behavior as the curve of the number of patterns. The gain constraint however has a different impact, indeed it does not reduce the execution times, the times oscillate around a same value. The third line presents results while varying the number of levels of the hierarchy. Indeed we modified the input hierarchy while deleting levels of abstraction from bottom to up, i.e., 0 levels of hierarchy means there is only $\mathscr{A}ll$ with all attributes of the dataset as children, while 5 levels of hierarchy means the normal hierarchy (presented in appendix A, p.131). We can see that the depth of the hierarchy has an effective impact on the execution. The number of patterns is reduced drastically, mainly between the depths 0 and 1. This shows that the use of a hierarchy has an important impact as a hierarchy of depth 0 is equivalent to no use of hierarchy. The execution times also reduce noticeably with the use of the hierarchy but is similar whatever the depth of the hierarchy.

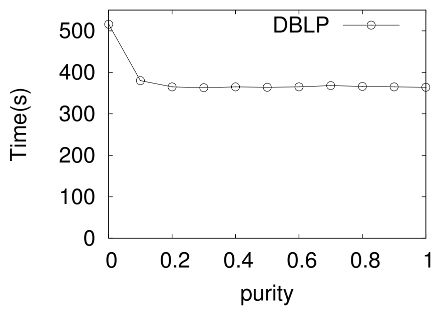
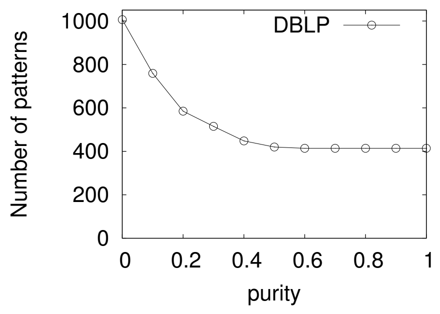
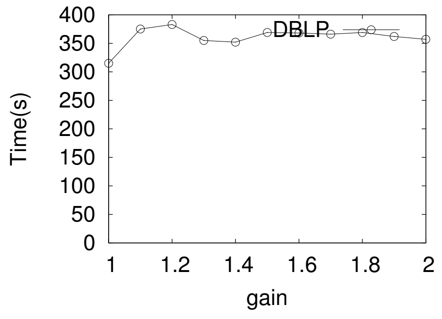
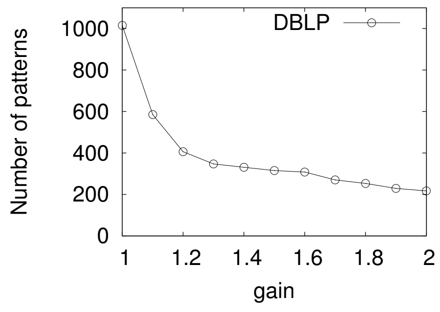
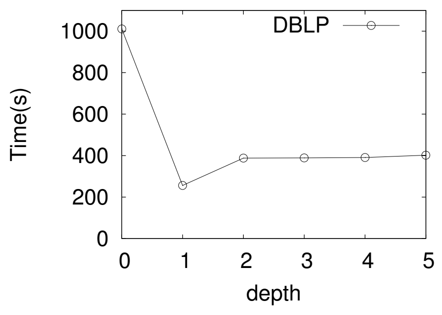
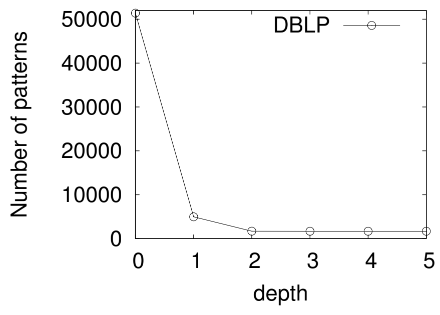
	Execution time (s)	Number of patterns
purityMin		
gainMin		
hierarchy depth		

Figure 3.19: Variation of ψ and γ and of the depth of the hierarchy.

We can also notice that even if the threshold are not really binding, the execution times are always acceptable. Indeed, the diameter threshold is set to 1 and running times would increase with a larger Δ.

Qualitative experiments

For this experiment, parameters were set to $\vartheta = 20$, $min_V = min_T = min_A = 2$, $\Delta = 2144$, $\gamma = 1.1$, $\psi = 0.35$ $\kappa = 0.2$ and $\tau = 0.4$. As this dataset has many attribute values equal to 0, it is not relevant to set the purity threshold too high. Considering the hierarchy, attributes too generalized as "conference" or "journal" are not really interesting, then γ was set to 1.1 to obtain patterns not too generalized. Two patterns were obtained in this extraction. The first pattern is presented in figure 3.20. This pattern concerns 17 authors who decreased their number of publication in VLDB and ICDE between 2004 and 2012. This pattern is relatively sparse, as the edges are dotted when they exist only at one of the two timestamps, for instance "Raymond T. Ng" is connected to "Beng Chin Ooi" at the first timestamp and to "Yannis E. Ioannidis" at the second timestamp, but he is connected to no author at both timestamps. We can say it represents small groups of authors who work together occasionnaly. Moreover, if the decreasing of publication in VLDB seems logical considering the new publication policy of the "VLDB endowment" it is noteworthy that it is also true for the ICDE conference. This pattern has small outside densities with $specificityV = 0.126$ and $dynamicT = 0.118$. As the decreasing in "VLDB" concerns many authors at this timestamp, we can say that the vertex specificity is mainly due to the decreasing in "ICDE". The small temporal dynamic point that they do not decrease their number of publication in these conferences at other timestamps and that the pattern can show that this small community change its publication policy.

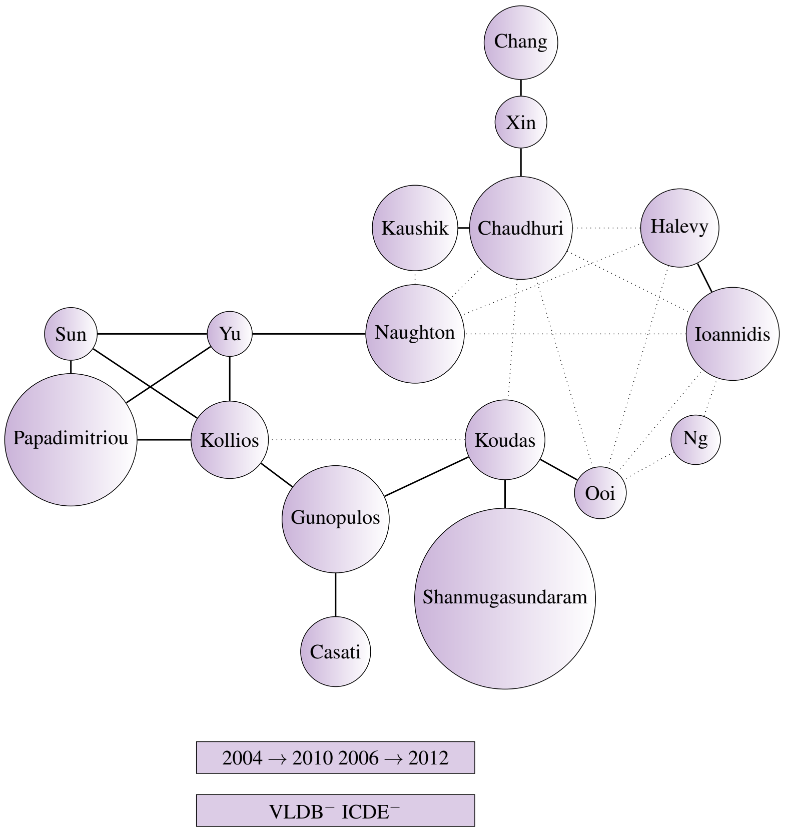


Figure 3.20: First pattern extracted from DBLP with the parameters: $\vartheta = 20$, $min_V = min_T = min_A = 2$, $\Delta = -1$, $\gamma = 1.1$, $\psi = 0.35$ $\kappa = 0.2$ and $\tau = 0.4$.

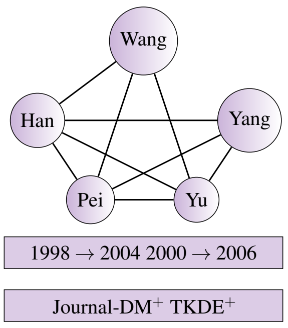


Figure 3.21: Second pattern extracted from DBLP with the parameters: $\vartheta = 20$, $min_V = min_T = min_A = 2$, $\Delta = -1$, $\gamma = 1.1$, $\psi = 0.35$ $\kappa = 0.2$ and $\tau = 0.4$.

The second pattern is presented in figure 3.21. It concerns 5 authors that increase their number of publication in the journal "IEEE-TKDE" and in the data-mining journals between 1998 and 2006. This pattern reflects that even if the journal "IEEE-TKDE" is considered as a databases journal in the hierarchy, it has a high attractivity in data-mining. The pattern has a purity of 0.417, which means that they publish in a lot of data-mining journals; it seems logical as these authors are well-known in the data-mining community. The vertex specificity is equal to 0.073 which depicts that this behavior is truly specific to these authors. And the temporal dynamic is equal to 0.4 which shows that their number of publication maybe oscillate. That point that it is difficult to publish regularly in this type of journals.

3.3 Skylines Of Hierarchical Co-Evolution Patterns

Setting all the thresholds is an issue itself. First, defining the thresholds may be arbitrary and lead to either a null or a very large number of patterns , which reduces the interpretation of the collection of patterns. For instance, the purity of the extracted patterns is really depending on the dataset, it is often greater than 0.9 for the "domestic US flights datasets"[2], but 0.3 is considered as a good value for patterns extracted from the "DBLP dataset". Another example is the threshold of vertex specificity which is difficult to set as the vertex specificity of the co-evolution patterns is variable. Second, given the number of thresholds to set, it is not obvious that the patterns that respect only some of them are not relevant. For instance a co-evolution pattern with many signed attributes and many timestamps has often few vertices. The use of a skyline of co-evolution patterns is proposed to apply better selection mechanisms.

The use of skylines or pareto dominance has been introduced as database operator[16] and recently studied in few works in data mining [59, 69, 88, 96]. Two papers are interested in graph analysis. Papadopoulos et al.[80] discover subgraph that are on the skylines considering 2 parameters, the number of vertices and the edge connectivity. Similarly, in [91], the authors adapt the framework of subdue method to the extraction of patterns that validate the Pareto dominance on two to three measures. The two last methods however use at most 3 parameters for the skyline. We propose to discover skyline patterns considering a multidimensional space composed with a subset of the measures proposed in section 3.2.

3.3.1 Definitions And Sky-H-MINTAG Algorithm

Given the measures $\mathscr{M} = \{sizeV, sizeT, sizeA, volume, purity, vertexSpecificity, temporalDynamic\}$, the skyline patterns are computed on a chosen subset $M \subseteq \mathscr{M}$ of these dimensions. The skylines are based on a pareto dominance which needs to be defined on these measures. The higher the values on the measures of size, purity and volume, the more relevant the patterns. However the lower the values on the measures of vertex specificity and temporal dynamic the more relevant the pattern.

A pattern P dominates another pattern Q with respect to a set of measures M if it dominates it on at least one measure $m \in M$ and dominates it or is equal on all measures. Formally:

Definition 3.3.1 — Pareto dominance. Given a set of measures $M \subseteq \mathscr{M}$, a pattern P dominates a pattern Q denoted $P \succ_M Q$, iff $\forall m \in M$, $m(P) \geq m(Q)$ and $\exists m \in M$ s.t. $m(P) > m(Q)$.

R We denote $m(P) > m(Q)$ (resp. $m(P) \geq m(Q)$) the relation of domination of a measure m, i.e., if $m(P) > m(Q)$ the value of P is considered as more relevant as the value of Q. However for some measures that must be minimized (in this work the measures of vertex specificity and temporal dynamic), the relation represents $m(P) < m(Q)$ (resp. $m(P) \leq m(Q)$). An example of a skyline of patterns using such minimized measures is proposed in Figure 3.22.

[2]"domestic US flights datasets" is a set of datasets used in Part III of the manuscript, see Appendix A for more details.

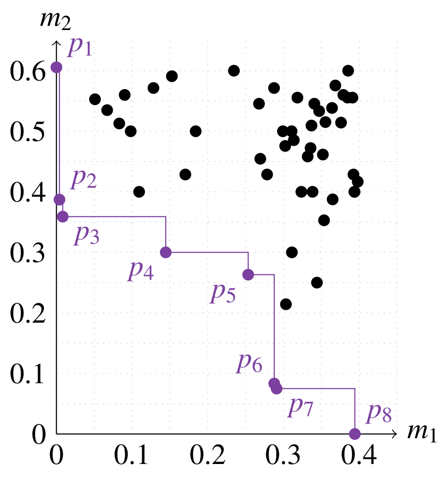


Figure 3.22: Plot of the extracted patterns w.r.t. their value of vertex specificity and temporal dynamic. Patterns of the skyline are coloured and linked together.

Once the dominance is defined, the idea is to extract all the patterns that are not dominated by any other pattern. Given a set of measures M and a set of patterns $\mathscr{P}$, a skypattern of $\mathscr{P}$ with respect to M is a pattern not dominated in $\mathscr{P}$ with respect to M. Formally:

Definition 3.3.2 — Skyline. The skyline operator $sky(\mathscr{P},M)$ returns all the skypatterns:

$$sky(\mathscr{P},M) = \{P \in \mathscr{P} | \nexists Q \in \mathscr{P} \text{ s.t. } Q \succ_M P\} \tag{3.9}$$

Given a user-defined set of measures $M \subseteq \mathscr{M}$, the aim is then to extract the complete set of patterns that respect the *diameter*, the *coevolution*, the *gain* and the *maximality* constraints, the constraints on the measures of $\overline{M} = \mathscr{M} \setminus M$ and which are on the skyline with measures M as dimensions.

Sky-H-MINTAG algorithm

The Algorithm 2 (p.84) needs small changes to use skylines, the adaptation is presented in Algorithm 3. As input of the algorithm the user must choose a set of constraints $\mathscr{C}$ among $(coevolution, diameter, sizeMinV, sizeMinT, sizeMinA, volumeMin, maximality, specificityV, dynamicT, purityMin)$ and define thresholds, and he must also choose a set of measures M to use with the skylines among $(sizeV, sizeT, sizeA, volume, vertexSpecificity, temporalDynamic, purity)$.

To prune the search space thanks to the skyline we define a predicate $relaxed\text{-}sky(P,C,M)$ which is true if patterns created from P and C can be in the skyline, i.e., if there is not a pattern already found that dominate them. To this aim we use the upper and lower bound defined on the measure of M. Formally, if $\exists$ a pattern Q such that $\forall m \in M$, $m(Q) \geq bound\text{-}m(P,C)$ and $\exists m \in M$ s.t. $m(Q) > bound\text{-}m(P,C)$, then no skyline pattern can be created from P and C.

An example of an execution of the Sky-H-MINTAG is proposed in Figure 3.23. The graph used is the toy example presented in Figure 3.10 (p. 78). The diameter threshold is set to $\Delta = 4$, volume threshold is set to $\vartheta = 2$, purity threshold is set to $\psi = 1$ and gain threshold is set to $\gamma = 1$; $sizeV$, $sizeT$ and $sizeA$ are used as measures of the skyline and other thresholds are set in such a way they do not constrain the patterns, i.e., $\kappa = \tau = 1$. To keep the figure readable, only a part of the enumeration is presented, the other parts are represented by dots.
Some nodes are noted “Not Pure Enough” or “Not Maximal” because of purity and maximality

Algorithm 3: Sky-H-MINTAG

```
Input: $P = \varnothing, C = (\mathscr{V}, \mathscr{T}, children(\mathscr{A}ll) \times S), D = \varnothing, attr, \mathscr{C}, M$
begin
    if $\neg C.empty$ then
        if $\mathscr{C}(P,C) \wedge relaxed\text{-}sky(P,C,M)$ then
            if $\neg(attr = \varnothing)$ then
                $child \leftarrow children(attr)$
                for $i$ in $1..|child|$ do
                    if $gainMin(P.V \cup C.V, P.T \cup C.T, P.A \setminus attr \cup child[i])$ then
                        Sky-H-MINTAG($(P \setminus attr) \cup child[i]$, $C \cup child[i+1..|child|]$,
                        $D \cup child[1..i-1]$, $child[i]$, $\mathscr{C}$)
                        $hasSon \leftarrow true$
                if $hasSon$ then
                    for $i$ in $1..|C.A|$ do
                        Sky-H-MINTAG($P \cup C.A[i]$, $C \setminus C.A[1..i]$, $D \cup C.A[1..i-1]$, $i$, $\mathscr{C}$)
                else
                    $attr \leftarrow \varnothing$
            if $attr = \varnothing$ then
                $E \leftarrow ElementTypeToEnumerate(P,C)$
                for $i$ in $1..|C.E|$ do
                    if $E = A$ then
                        $attr \leftarrow C.E[i]$
                    Sky-H-MINTAG($P \cup C.E[i]$, $C \setminus C.E[1..i]$, $D \cup C.E[1..i-1]$, $attr$, $\mathscr{C}$)
                Sky-H-MINTAG($P$, $C \setminus C.E$, $D \cup C.E$, $\varnothing$, $\mathscr{C}$)
    else if $\mathscr{C}(P) \wedge sky(P,M)$ then
        output ($P$)
```

constraints. We are interested here in the nodes denoted with "Dominated by". The most interesting node is the crossed one, i.e., the pattern P_1. It has been developed and considered as a valid pattern as it respects the constraints of diameter, volume and purity and as it was not dominated by any pattern in the skyline (there was no pattern yet). Later in the execution, the pattern P_3 has been found with 3 vertices, 1 time and 1 attribute, and dominates P_1 which has 2 vertices, 1 time and 1 attribute, then pattern P_1 is deleted from the skyline.

3.3.2 Empirical Study

Let us now report on experimental results, using Algorithm 3, to illustrate the interest of the proposed approach by varying M, $\mathscr{C}$ and the different thresholds.

Quantitative experiments

Some experiments were done to evaluate the impact of the parameters on the number of patterns and the execution times while using skylines. As there is many parameters and the possibility to use them or not as a dimension of the skyline, it is difficult to present all the results. However, Figures 3.24, 3.25 and 3.26 present for each parameter $sizeV$, $sizeT$, $sizeA$, $specificityV$, $dynamicT$, $volume$ and $purity$ an aggregation of the results obtained with the other measures used as a threshold constraint or as a dimension of the skyline. Each graph presents the minimal values, the average values and the maximal values over 64 sets of experiments. An experiment was also done using all these measures as dimensions of the skyline, 358 patterns were extracted in 610 seconds. For all these experiments γ was set to 1.1 and Δ was set to 1, and when fixed other parameters were set to $min_V = min_T = min_A = 2$, $\vartheta = 5$, $\psi = \kappa = \tau = 0.2$.

Figure 3.24 presents the results over the size measures, i.e., $sizeV$, $sizeT$ and $sizeA$. The

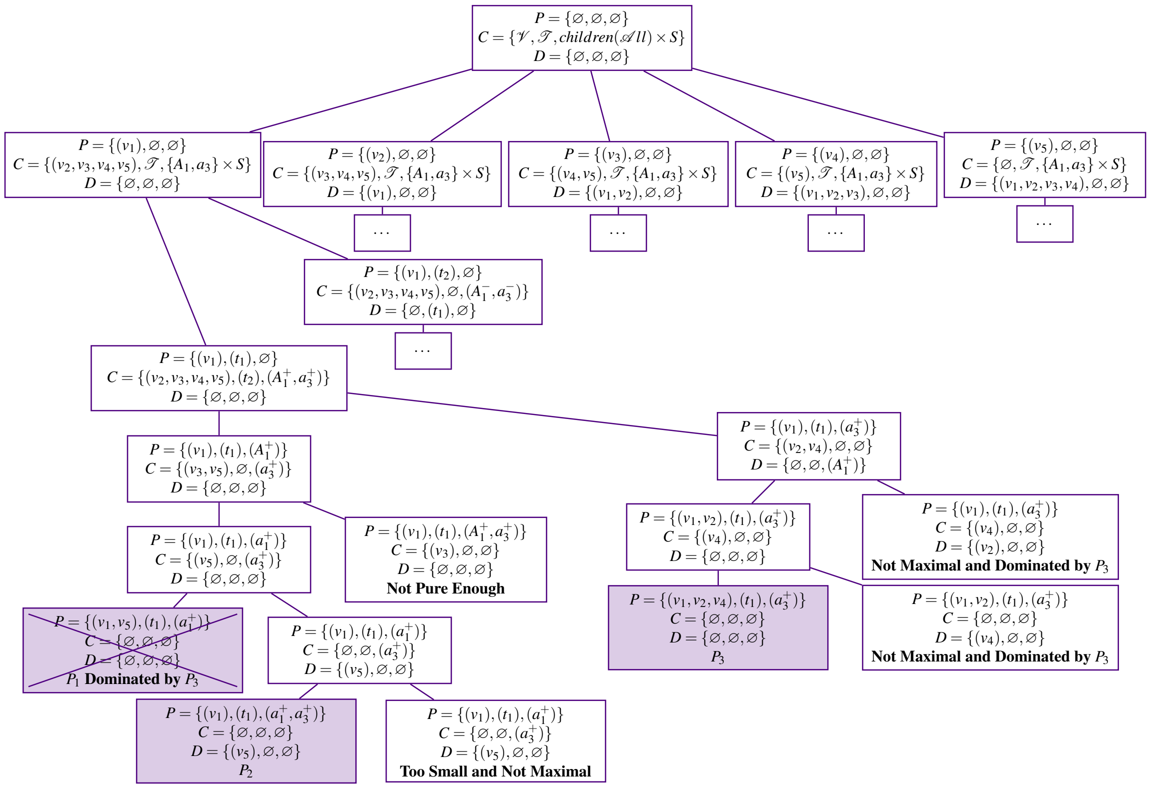

P = {∅,∅,∅} C = {𝒱, 𝒯, children(𝒜ll) × S} D = {∅,∅,∅}
P = {(v1),∅,∅} C = {(v2,v3,v4,v5), 𝒯, {A1,a3} × S} D = {∅,∅,∅}
P = {(v2),∅,∅} C = {(v3,v4,v5), 𝒯, {A1,a3} × S} D = {(v1),∅,∅}
P = {(v3),∅,∅} C = {(v4,v5), 𝒯, {A1,a3} × S} D = {(v1,v2),∅,∅}
P = {(v4),∅,∅} C = {(v5), 𝒯, {A1,a3} × S} D = {(v1,v2,v3),∅,∅}
P = {(v5),∅,∅} C = {∅, 𝒯, {A1,a3} × S} D = {(v1,v2,v3,v4),∅,∅}
P = {(v1),(t2),∅} C = {(v2,v3,v4,v5),∅,(A1−,a3−)} D = {∅,(t1),∅}
P = {(v1),(t1),∅} C = {(v2,v3,v4,v5),(t2),(A1+,a3+)} D = {∅,∅,∅}
P = {(v1),(t1),(a3+)} C = {(v2,v4),∅,∅} D = {∅,∅,(A1+)}
P = {(v1),(t1),(A1+)} C = {(v3,v5),∅,(a3+)} D = {∅,∅,∅}
P = {(v1,v2),(t1),(a3+)} C = {(v4),∅,∅} D = {∅,∅,∅}
P = {(v1),(t1),(a3+)} C = {(v4),∅,∅} D = {(v2),∅,∅} Not Maximal and Dominated by P3
P = {(v1),(t1),(a1+)} C = {(v5),∅,(a3+)} D = {∅,∅,∅}
P = {(v1),(t1),(A1+,a3+)} C = {(v3),∅,∅} D = {∅,∅,∅} Not Pure Enough
P = {(v1,v2,v4),(t1),(a3+)} C = {∅,∅,∅} D = {∅,∅,∅} P3
P = {(v1,v2),(t1),(a3+)} C = {∅,∅,∅} D = {(v4),∅,∅} Not Maximal and Dominated by P3
P = {(v1,v5),(t1),(a1+)} C = {∅,∅,∅} D = {∅,∅,∅} P1 Dominated by P3
P = {(v1),(t1),(a1+)} C = {∅,∅,(a3+)} D = {(v5),∅,∅}
P = {(v1),(t1),(a1+,a3+)} C = {∅,∅,∅} D = {(v5),∅,∅} P2
P = {(v1),(t1),(a1+)} C = {∅,∅,(a3+)} D = {(v5),∅,∅} Too Small and Not Maximal

Figure 3.23: Part of an execution of Sky-H-MINTAG.

constraints on the size of the set of vertices and of the set of times allow a great pruning of the number of patterns. However the constraint on the number of attributes does not. This can be explained as the more attributes in the pattern are, the less vertices or timestamps, then there is a lot of patterns with a same number of vertices or timestamps (e.g., patterns with only one vertex and one timestamp) which implies a lot of patterns that respect the skyline. As a logical consequence, the execution times are reduced by each one of these constraints. The most noticeable reduction is between $min_T = 1$ and $min_T = 2$. This is due to the fact than there is no continuity in most of the trends of this dataset.

Results while varying the threshold κ of vertex specificity and the threshold τ of temporal dynamic are described in Figure 3.25. The vertex specificity does not have a major impact neither on the number of patterns nor on the execution times. The temporal dynamic, however, allows a small reduction of the both.

Results over the measure of *volume* and the measure of *purity* are presented in Figure 3.26. Increasing the volume threshold ϑ allows a large reduction of the number of patterns, and even if the constraint is not anti-monotonic considering the algorithm, it allows a reduction of the execution times. The purity threshold also implies a reduction of the number of patterns, however there is nearly no decreasing of the execution times.

The behaviour of the algorithm with the skyline analysis is similar to the behavior of the algorithm without it (section 3.2.5, p.87).

Figure 3.27 describes the number of patterns and execution times using 0 to 6 dimensions for the skyline. Each graph presents the minimal values, the average values and the maximal values; γ was set to 1.1 and Δ to 1. For each number of dimensions $\{0,1,2,3,4,5,6\}$, there were respectively $\{50,300,750,1000,750,300,50\}$ experiments. Results show that the higher

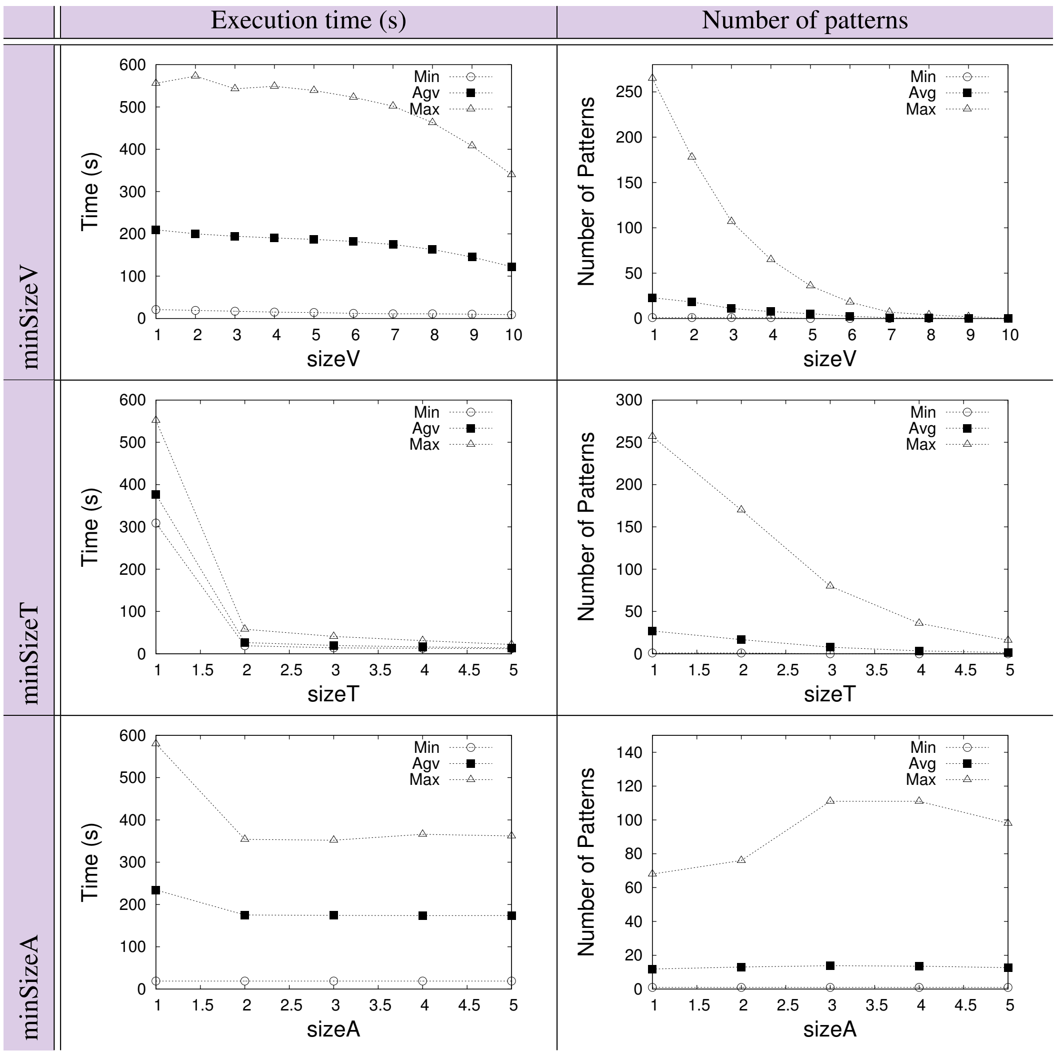


Figure 3.24: Variation of min_V, min_T and min_A.

the number of dimensions, the higher the execution times. However the execution times always stay under 11 minutes. Considering the number of patterns, we can see that the maximal number oscillates, but the average number merely increases at the same time as the number of dimension. Indeed there is a small decreasing between 0 and 2 dimensions (passing from 17.1 to 4.4) and then a continuous increasing between 2 and 6 (passing from 4.4 to 153.7).

We have seen than too many dimensions for the skyline may lead to a large number of patterns. Moreover some constraints imply a better pruning of the execution times together with a better reduction of the number of patterns. We then recommend to use between 2 and 4 dimensions in the skyline. For the constraints outside the skyline, we recommend *sizeV*, *sizeT* and *volume* as they are more easy to set for the user and they allow an efficient pruning. Constraints *sizeA* and *purity* can also be useful outside the skyline as the first one is easily fixable by the user and the second one allows an efficient pruning.

Qualitative experiments

Figure 3.28 depicts two patterns extracted with parameters $\Delta = 2, \gamma = 1, min_V = 5, min_T = 3, \psi = 0.3$ and other measures as skyline dimensions. As there is multiple timestamps thick edges represent relations that exist at each timestamp and dotted edges represent relations that exist at 1 or 2 timestamps. These two patterns concern exactly the same dates and trend, but distinct set

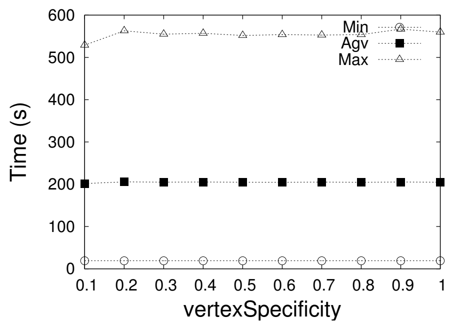
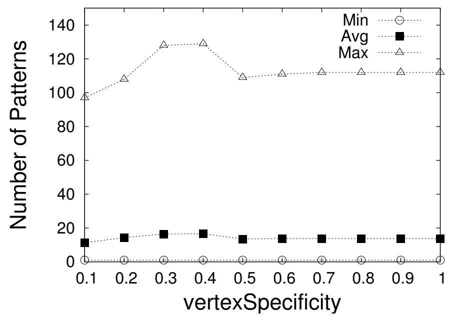
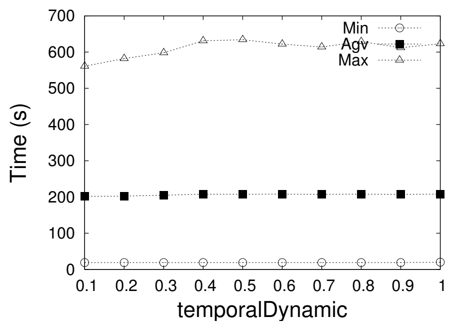
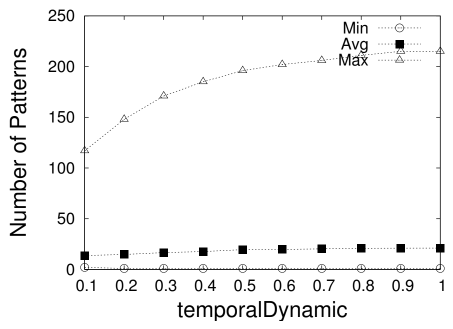
	Execution time (s)	Number of patterns
specificityV		
dynamicT		

Figure 3.25: Variation of κ et τ.

of authors. Even if they concern the same attribute and timestamps they have relatively small outside densities, with respectively $specificityV = (0.23, 0.23)$ and $dynamicT = (0.1, 0.17)$ which means there isn't so many authors that respect this trend and not so many timestamps where this trend is respected by the authors of the patterns. It is also noticeable that even if γ is small, the attribute has not been specialized and the purity is equal to respectively 0.33 and 0.35 which is a relatively good purity value for this dataset. Comparing the two patterns, we can also notice that the one on the left is more sparse than the one on the right, then, the pattern on the right represents a group of authors more used to work together.

If we launch one extraction with no skyline analysis and with thresholds $\kappa = 0.23, \tau = 0.17$, $\vartheta = 15$ and $min_A = 1$, only these two patterns are extracted. However, if we launch one extraction with no skyline and with thresholds $\kappa = 1, \tau = 1$, $\vartheta = 1$ and $min_A = 1$, seven patterns are extracted. These patterns are represented in Figure 3.29 with regard to their value of vertex specificity and temporal dynamic. It is noticeable than all these patterns have a similar vertex specificity (0.2273 and 0.2269), this can be explained by the fact that the timestamps and the attribute are the same for all patterns and they have either 5 or 6 vertices. However the temporal dynamic vary because the authors mainly change and they do not have a similar behaviour on other timestamps. Then, if the extracted patterns depict similar behaviours, the skyline highlights the authors for whom it is the most specific.

3.4 Conclusion

We proposed interestingness measures to compare the behaviour of the patterns with respect to the other vertices, the other timestamps and the other attributes. These measures allow the user to extract patterns that depict a more specific behaviour in the dataset. Experiments prove that the number of resulting patterns is greatly reduced and that extracted patterns are relevant.

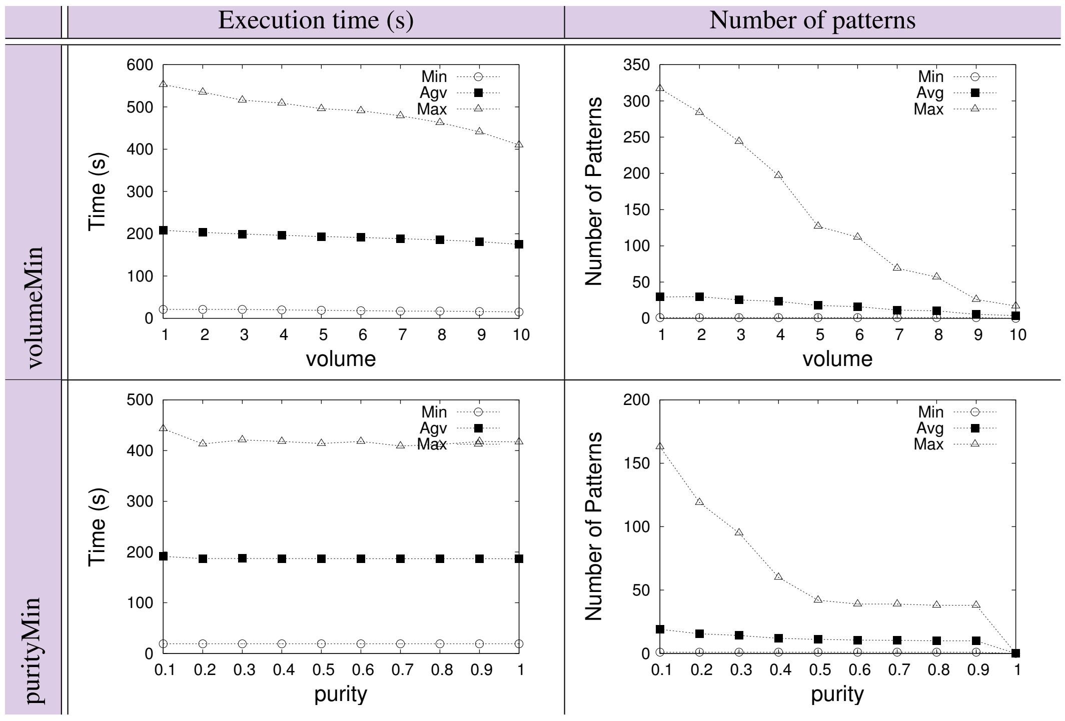


Figure 3.26: Variation of ϑ and ψ.

Moreover each measure provides an additional information on the pattern which makes possible to have a finer analysis of its meaning.
We also proposed the use of a hierarchy on the attributes to enable the extraction of hierarchical co-evolution patterns. Such patterns can describe both precise patterns, i.e., patterns with a set of attributes of the dataset, and general patterns, i.e., patterns with attributes that describe the behaviour of a set of attributes of the dataset. Experiments proved that it reduces the number of resulting patterns and lets the possibility to obtain larger sets and more general information while avoiding some redundancy in the collection of patterns.
We proposed the use of skyline of co-evolution patterns to both facilitate the use of the method by experts by reducing the number of parameters to set and extract the most relevant patterns of the collection with regard to users' preferences. Some thresholds are difficult to set and relevant

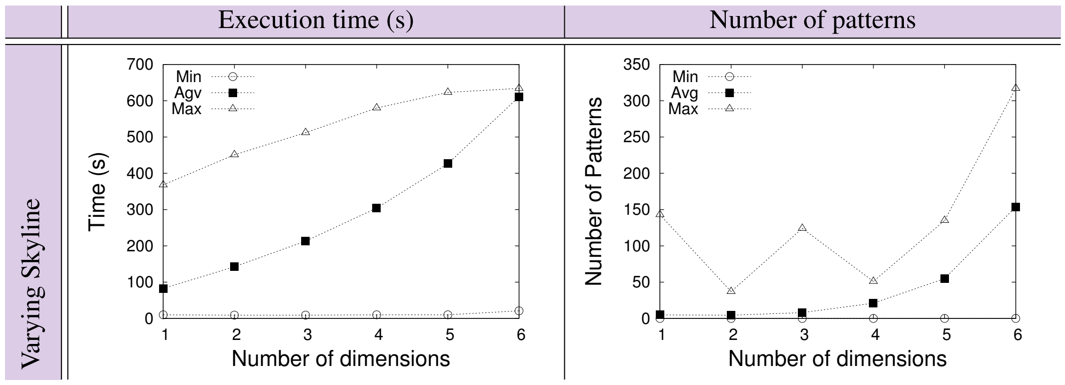


Figure 3.27: Number of patterns and execution time on the "DBLP" dataset while varying the number of parameters used as dimensions of the skyline.

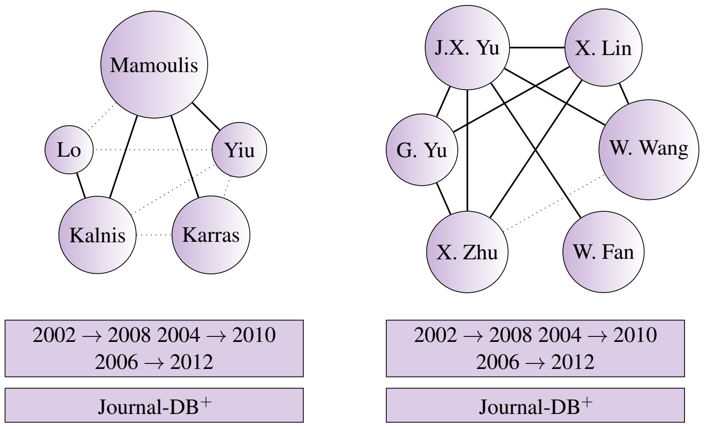


Figure 3.28: Patterns extracted from "DBLP" with $\Delta = 2, \gamma = 1, min_V = 5, min_T = 3, \psi = 0.3$ and other constraints as skyline dimensions.

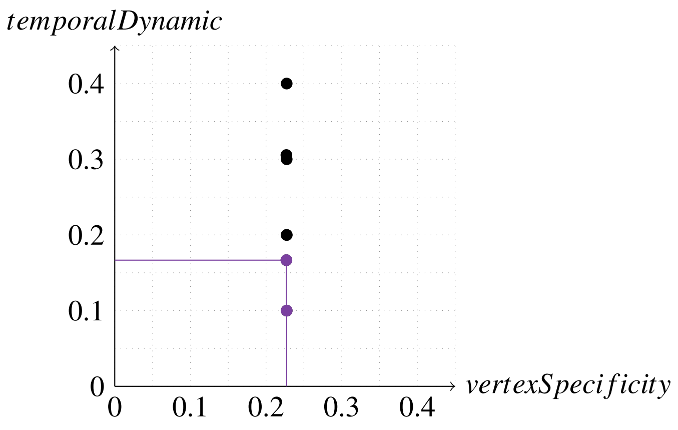


Figure 3.29: Plot of the extracted patterns w.r.t. their value of vertex specificity and temporal dynamic. Patterns of the skyline are coloured and linked together.

patterns can have great values on some measures but not all of them, then skyline analysis avoid to fix such thresholds and highlights the patterns which are not dominated by any other.
These three propositions imply a collection of resulting patterns easier to analyse for the expert. Considering the execution times, the specificity measures allow an effective reduction and even if the use of a hierarchy or of the skylines analysis can slightly increase the running times, they still stay reasonable. To assess the relevancy of the extracted patterns, there is a need of more qualitative experiments. To this aim, next part proposes experiments on several real-world datasets.

Part III

Application To Spatio-Temporal Data

Outline

We propose applications of co-evolution pattern mining on spatio-temporal data. On one hand, there is an increasing development of sensors, sensor networks, social networks etc. that produce traces continuously. On the other hand, geolocation tools are ubiquitous and all these data are more and more associated to geographical and spatial information. To get insights about underlying phenomena that one want to understand, describe, and/or control.

This work is funded by the "FOSTER" project whose main goal is to exploit satellite images to support erosion understanding. In this context, co-evolution patterns can describe specific behaviours of objects in the images, and give rise to interesting hypothesis to understand or describe erosion. To this aim, satellite images have to be transformed in dynamic attributed graphs and we propose several ways to do it. We also discuss the challenges risen by this kind of data on the extraction of relevant co-evolution patterns. Finally we provide preliminary experiments on two satellite images before and after landslides in Brazil.

To evaluate our work on another type of spatio-temporal dataset, we present experiments on four dynamic attributed graphs created from a public database on the domestic flights in the United States. The idea is to extract patterns that describe specific behaviour of a set of airports or extracting patterns that reflect events that impacted the domestic flights in the United States.

This part of the manuscript is organized as follows. In Chapter 4, we discuss the application of co-evolution pattern mining on satellite images and we propose a preliminary study. In Chapter 5, we evaluate the different methods defined in Part II on "domestic US flights" datasets and we discuss the results.

4 — Application To FOSTER Data

The FOSTER project aims at providing geologists tools for monitoring and better understanding the soil erosion. This process is based on multi-temporal very high resolution satellite images, sensor data and/or expert knowledge. We present how we could study such data with the methods presented in the manuscript, i.e., how we could transform images in spatio-temporal graphs and extract patterns that could describe erosion.

In this chapter we will discuss how satellites images could be transformed in dynamic attributed graphs and how the patterns could provide new insights to experts. We also discuss the limits and difficulties of the analysis of such images through our method. As a preliminary study, we propose some experimental results on the analysis of satellite images in Brazil before and after huge landslides.

4.1 From Satellite Images To Dynamic Attributed Graphs

First of all, we need to transform the data in dynamic attributed graphs. It is straightforward that timestamps are images ordered temporally and vertices are elements that have been identified in the images. Spatiality can be handled in two ways, indeed either edges can represent spatial relations as neighbourhood or attributes can represent spatial position as coordinates. In the context of the "FOSTER" project, we want to describe soil erosion, then, we are interested in the study of the characteristics of the identified elements of the image. While treating images, it seems then easier to use edges to describe spatial relations and attributes to depict information specific to the element. The spatial relation connects elements that evolve in a same way, i.e., whose characteristics follow the same trends.

Let us present how to transform images in graphs. First, in our model of dynamic attributed graphs, vertices do not change over time, i.e., a vertex is a same object over time. A satellite image is a set of pixels that have coordinates, then it is easy to study the evolution of a same pixel over time. In the domain of satellite images analysis, it is frequent to segment the image into shapes, i.e., segmented areas in the image, however, this segmentation is often done for each image, then there is a different set of segments for each timestamp. To use segments as vertices, either the segmentation must be done for all the images, or segments must be related through time. The attributes of the graph are then spectral responses in red, green, blue, near infra-red, etc., of the image and other measures computed from these values. A last possibility would also be to identify object in the images that we can follow over multiple timestamps, as for instance geological faults. These objects could then be vertices of the graph and characteristics of the objects as the size, the colors, etc. can then be associated as attributes of each object.

Here are three examples of possible dynamic attributed graphs:

1. Given n images with a segmentation of s segments common to all timestamps. At each timestamp m measures are computed on each segment: $infra\text{-}red$, red, $blue$, $green$, $NDVI$, $RedIndex$, etc., which are the average pixel values. All these attributes provide information that can be interpreted to quantify the amount of vegetation, the amount of erosion or to identify the type of soil for instance. The dynamic attributed graph is a sequence of n attributed graphs where each vertex is a segment, there is an edge between two vertices if the two segments are neighbours and each vertex at each timestamp is associated to the m attribute values. In this example the structure does not change.
2. Given n images with a segmentation of s_i segments specific for each timestamp i created with one of the many image segmentation techniques [77] as for instance the multi-resolution segmentation [7] used to segment the images of Brazil presented in this chapter. At each timestamp m measures are computed on each pixel. The dynamic attributed graph is a sequence of n attributed graphs where each vertex is a pixel, there is an edge between two vertices if the pixels are either neighbours or if they are in the same segment at this timestamp, and each vertex at each timestamp is associated to the m attribute values. Even if the neighbourhood of each pixel does not change, it is taken into account to enable a connected graph instead of a set of connected component (one for each segment).
3. Given n images where experts have identified o objects common to nearly all the images. At each timestamp, m measures are computed on each object. The dynamic attributed graph is a sequence of n attributed graphs where each vertex is an object, there is an edge between two vertices if the two objects at a distance lower than a given threshold in the image and each vertex at each timestamp is associated to the m attribute values. As objects can appear or disappear from the image, they are not connected to other vertices at these timestamps.

In such domains, there is often associated expert knowledge, and the aim of using data mining techniques is either to help the expert or to discover additional information. Co-evolution patterns could represent events that appear suddenly in some areas of the image as for instance a landslide. They could also identify "objects", i.e., sets of segments or pixels that have a same evolution through time as for instance a lavaka which is a type of erosional feature. This could be used by the expert to help classify elements in an image, validate a classification, confirm contours of such regions. This could also be used to identify new attributes that co-evolve with known one during an erosion.

However the study of satellite images implies some important difficulties. One major limit considering our method is the scalability. Indeed, nowadays, images are of high quality and each image is easily composed of hundreds of thousands of pixels. For our proposed methods it is difficult to scale such an amount of data. However the use of user knowledge to constrain the pattern could reduce enough the search space to extract interesting information. Another limit of this type of data is technical problems such as alignment of the images and change of luminosity. Indeed, problems of alignment can associate through time pixels that do not represent exactly the same area. This can imply extraction of trends that do not relate real evolutions but only a change of area that is associated to this identifier of pixel. Second, images are taken at different dates and hours, then luminosity can be really different depending on the weather, the season, the hour of the day etc. This change of luminosity implies a change of spectral response and then a variation in the attribute values that can cover the trends to extract and complicate pattern extraction.

In the next section, we propose a preliminary study on satellite images provided by the FOSTER project. To transform the images in dynamic attributed graphs we chose the first method, a schematic example is proposed in Figure 4.1.

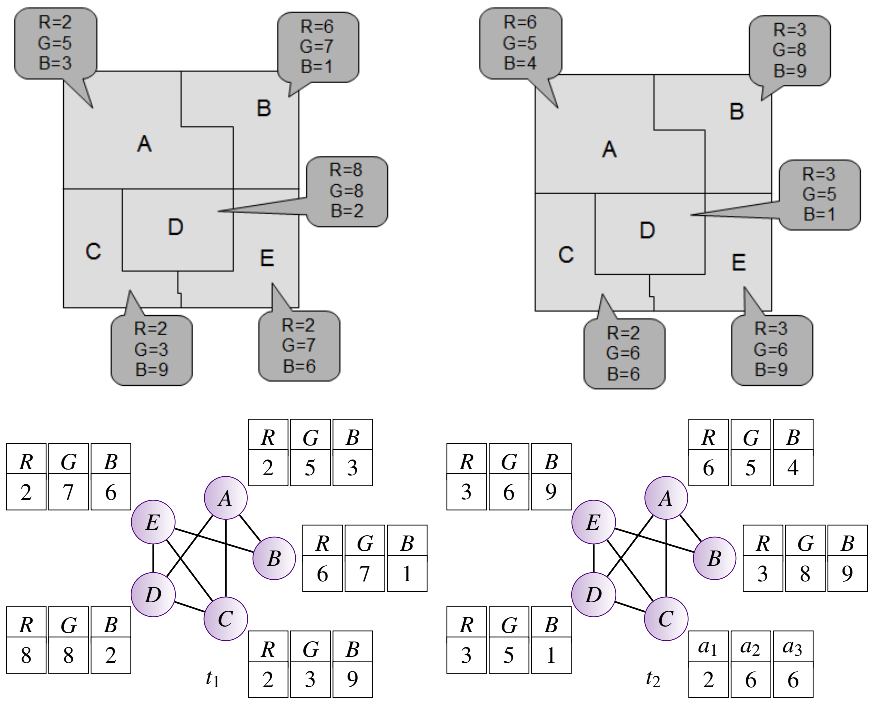


Figure 4.1: Example of transformation of the images in a graph (Option 1).

4.2 Study Of "Brazilian landslides" Dataset

In January 2011, huge landslides happened in Brazil after a week of almost continuous rain. These landslides are large enough to be visible from satellite images. The dataset "Brazil" is derived from 2 satellite images taken before and after the landslides. The image after the landslides is presented in figure 4.2.

The dynamic attributed graph is composed of $394,885$ vertices that stand for image shapes (segmented areas). There is an edge between two vertices if the corresponding shapes are contiguous. Each segment is associated to 10 attributes that are the spectral response in infra-red (NIR), red (R), blue (B), green G) and indices computed from these values, $\frac{G}{R}$, $\frac{B}{G}$, $\frac{R}{NIR}$, NDVI (i.e., $\frac{NIR-R}{NIR+R}$), red index (i.e., $\frac{R^2}{B\times G^3}$) and brilliance index (i.e., $\sqrt{NIR^2+R^2}$). To summarize, this dataset consists in $394,885$ vertices, 2 timestamps and 10 attributes. More details on the dataset can be found in appendix A.

As explained previously, this dataset implies two major difficulties:
First it is rather large, there is not many timestamps and attributes but $394,885$ vertices is a challenge for our method. Moreover the structure is rather "regular", indeed, even if there are

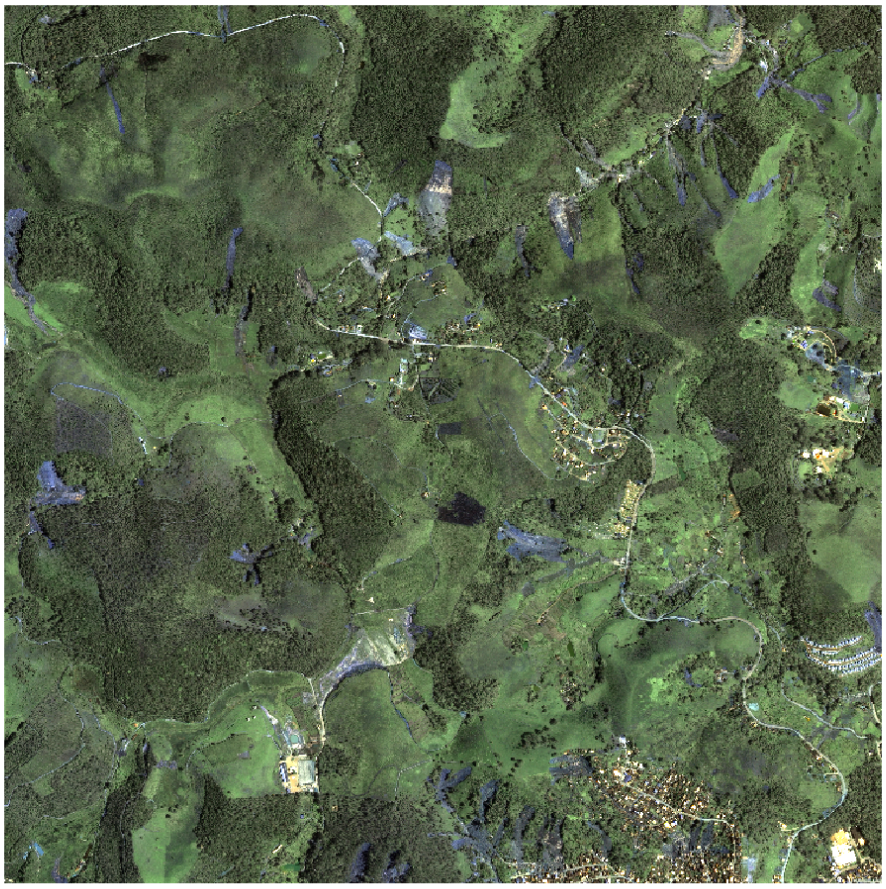


Figure 4.2: Satellite image taken after the landslides of January 2011 in Brazil.

segments of different size, it could be compared to an irregular mesh, and it is one large connected component. Then, the measures on the neighbourhood can not have an impact significant enough to reduce drastically the search space. Using similarity, we then took only an extract of the dataset as it couldn't scale the entire images and using diameter we choose $\Delta = |\mathcal{V}| - 1$ which reduces drastically the computation times.
Second, the two images were taken at different seasons and different hours which impact the colour and then our attribute values. This change of context has a similar impact on the evolution of the attribute values of each pixel. That's why we use for the experiments 0.7-strictEvol (2.2.3, p.47), i.e., we consider evolutions if they are of at least 70% of the value of the attribute ($\delta = 0.7$).

The aim of the extraction using this dataset is to find patterns that describe landslides, which could be simplified as finding erosion. Three of the attributes are interesting erosion indices, especially NDVI. NDVI is a vegetation index, the higher the NDVI value, the more there is vegetation. As a landslide destructs vegetation, the decreasing of the NDVI value can be very useful to highlight landslides. To help reduce the execution time, we then choose to only extract patterns including a decreasing NDVI value, i.e., only patterns containing $NDVI^{-}$ were enumerated.

4.2.1 Mining Patterns With Similar Vertices

Since the algorithm using similarity did not scale on the entire dataset, we created a smaller dataset selecting 10,521 shapes in the north west of the image. This part of the image contains at least three visible part of landslides. We ran experiments with the Jaccard similarity ($\sigma = 0.5$) to discover sets of vertices that are in a similar neighborhood and we set $\delta = 0.7$. We also forced the patterns to contain the attribute $NDVI^-$ and obtained 297 patterns whose NDVI decreased between the two first images, i.e., after the landslides. Many patterns overlap and all the patterns can be regrouped in 5 major sets. The shapes of the patterns and the sets are represented in white in Figure 4.3.

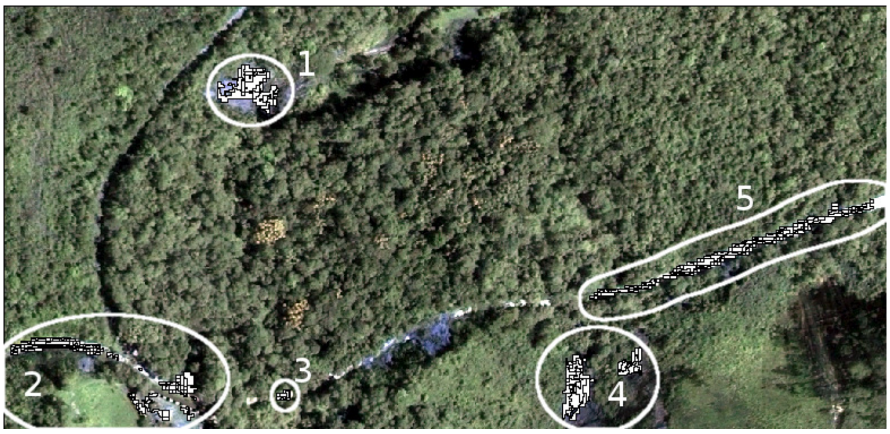


Figure 4.3: Shapes from patterns extracted using Jaccard similarity.

The sets of regions 1 and 4 are actual landslides, all the pixels corresponding to a landslide are not found, however a major party of them is. The Groups 2, 3 and 5 are not landslides, when looking at the original data, we can see a shift between the two pictures. Indeed, they are not perfectly calibrated and looking at the first image, the sets of regions 2, 3 and 5 correspond to vegetation boarding the road, whereas looking at the second image, they correspond to the road. One can also notice that there is a small landslide in the bottom center of the image that is not found.

All these patterns are rather small and many patterns are necessary to cover a small landslide, indeed, Jaccard similarity implies that shapes must have at least 50% of common neighbourhood. Considering that a landslide is composed of many segments, it is obvious that some have no common neighbourhood.

4.2.2 Mining Patterns With A Maximum Diameter

Here, we also focus on patterns that involve $NDVI^-$, with a diameter $\Delta = |\mathscr{V} - 1|$, i.e., looking for connected component, which seems more adapted to the search of large sets of vertices. Using such parameters, the algorithm scales on the entire dataset and returned 2121 patterns in 3 hours, that involve 42072 regions reported on Figure 4.4.

These results were evaluated by an expert who certified that 69% of the shapes that are considered as a true landslide appear in the computed patterns, i.e., we found a large majority of the landslides in this extraction. Considering the extracting segments, 46% correspond to "landslides" while the rest does not. The 54% remaining regions can be classed in the 4 following categories:

1. *Regions nearby true landslides which have not been interpreted as landslides by the expert.* Looking more precisely at the picture, the red regions are often surrounded by small white ones which we consider mainly as "border effect". Indeed, first the classification as "landslide" or "not landslide" was made by an expert looking at the picture, however two experts can make different choices on a same segment. Second, the vegetation around

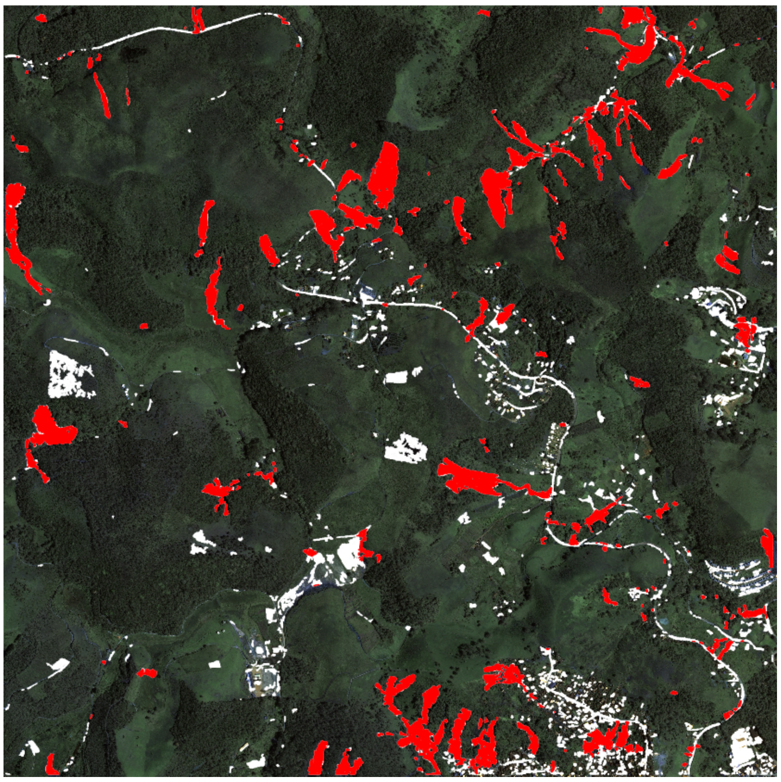


Figure 4.4: Shapes involved in the patterns: true landslides (red) and other phenomena (white).

a landslide can also be damaged and if the respective segments can not be considered as landslides, yet, the NDVI decreased.

2. *Deforested area not due to landslides.* When comparing the two images, some patterns correspond clearly to human activity. For instance, the large white region on the middle left of the picture. Before the landslide there were a forest, after the landslide there is no tree anymore.
3. *Regions found due to misalignment of the segmentation technique.* We explained in last section that the two images are not perfectly aligned. This causes some patterns to be extracted, as for instance the main road that cross the image from top left to bottom down.
4. *Regions that represent cities and human activity footprints.* Cities and other human constructions are indeed almost all found, some of them are easily recognizable in the picture. There is no vegetation in this regions, however the spectral response in red and infra-red seems to imply a decreasing of the NDVI due to the change of luminosity of the two images.

4.2.3 Discussion On The Results

Figure 4.5 presents a zoom in the section of the image selected for the extraction with 10,521 vertices in section 4.2.1 (p.108). The shapes presented are those of patterns extracted using the diameter constraint in section 4.2.2 (p.109). It is noteworthy that the extracted shapes are quite similar. One can recognize the 5 groups of patterns, however using diameter, the landslides are better covered and some new small patterns are found in the bottom center of the image.

The two datasets have 10 attributes and 2 timestamps, however the small one contains only 10,521 vertices, i.e., around 40 times less. Using similarity, 297 patterns are extracted from the small dataset, while 2121 are extracted from the large dataset using the diameter constraint, i.e., proportionally less that five times less. However, as we can see in Figure 4.5, the same

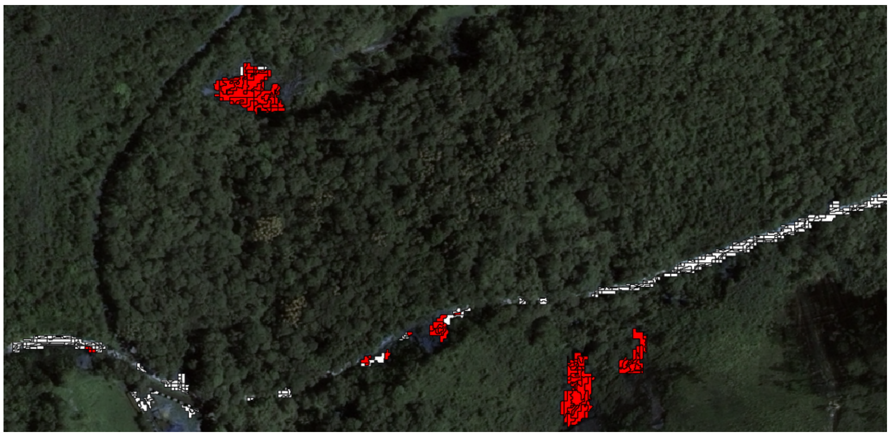


Figure 4.5: Shapes involved in the patterns found using diameter in the small subset of the image used for the experiments presented in section 4.2.1 (p.108).

regions are extracted. That can be explained by the fact that patterns are larger using the diameter constraint with $\Delta = |\mathscr{V} - 1|$, i.e., while extracting connected component. Indeed, the similarity measure is not transitive, then segments from one end of the landslide to the other end do not have a similar neighbourhood and a landslide is depicted by several patterns. With the extraction of a connected component, one landslide can be represented by one pattern.

Using such a dataset, it is then more interesting to use the diameter as a structural constraint to extract larger patterns that better coincide with the objects we want to study.

5 — Application To “Domestic US Flights”

Four dynamic graphs about US flights were generated with aggregated data over different periods of time of the RITA “On-Time Performance” database[1]. This database contains on-time arrival data for non-stop US domestic flights by major air carriers. Graph vertices stand for US airports and are connected by an edge if there is at least a flight connecting them during the time period. We consider 8 vertex attributes that are the number of departures/arrivals, the number of cancelled flights, the number of flights whose destination airport has been diverted, the mean delay of departure/arrival and the ground waiting time departure/arrival. More information can be found in appendix A. The four dynamic graphs are:

- **Last 20 years:** Data are aggregated over each year between 1992 and 2011: $|\mathscr{V}| = 361$, $|\mathscr{T}| = 20$, $|\mathscr{A}| = 8$
- **Two years around 9/11:** Data are aggregated over each month between September 2000 and September 2002: $|\mathscr{V}| = 234$, $|\mathscr{T}| = 25$, $|\mathscr{A}| = 8$
- **September 2001:** Data are aggregated over each day of September 2001: $|\mathscr{V}| = 220$, $|\mathscr{T}| = 30$, $|\mathscr{A}| = 6$
- **Katrina:** Data are aggregated over each week between 08/01/2005 and 09/25/2005: $|\mathscr{V}| = 280$, $|\mathscr{T}| = 8$, $|\mathscr{A}| = 8$

The September 11 attacks had a major impact in the USA especially on the aviation sector. The aim of the datasets “Two years around 9/11” and “September 2001” is to evaluate the impact on a short and a long term on the US domestic flights. Hurricane Katrina was the deadliest and most destructive Atlantic hurricane of the 2005 Atlantic hurricane season. It was the costliest natural disaster, as well as one of the five deadliest hurricanes, in the history of the United States. Among recorded Atlantic hurricanes, it was the sixth strongest overall. In the experiment with the dataset “Katrina”, we aim to characterize the impact of this hurricane on the US domestic flights.

The structural specificity of these datasets is the small diameter and the rather high density. This can impact the computation of structural constraints either on the computation times and on the reduction of the number of patterns. The semantic specificity of this dataset is the regularity of the trends. Indeed except special events, the dataset is based on an organized flights schedule and the number of flights can increase or decrease regularly throughout days of the week, seasons,

[1] http://www.transtats.bts.gov

holidays, etc. A major challenge is then to extract interesting trends in all these known trends.

5.1 Mining Co-Evolution Patterns With Regard To Similarity Or Diameter

Let us first see how does our simple algorithm presented in chapter 2 react on these datasets. To this aim we discuss some results using different thresholds.

Last 20 years

First, let us extract patterns using the Jaccard similarity. As the dataset is dense, we set the threshold σ to 0.5 (minimum similarity, p.47), i.e., the airports in the pattern must have at least half their neighbors in common. We also constrain the volume to $\vartheta = 70$ (minimum volume, p.51) as the aggregation by years implies a lot of regular trends and then size of patterns that are easily large. Seven patterns were found in 22 seconds with six containing an overlapping set of vertices. The airports of these patterns are presented in figure 5.1.
Six patterns are presented in the picture on the left, with different shapes and colors for each pattern. All these patterns concern a subset of the time steps 1993-1994, 1994-1995, 1995-1996, 1996-1997, 1998-1999, 1999-2000, 2003-2004, 2005-2006, 2008-2009, 2010-2011 and a subset of the attributes "nbDep","nbArr","nbCan","depDel","arrDel" that increase. The picture on the right presents a pattern where the number of departures and the number of arrivals in airports increase between 1992 and 1994, between 1996 and 2001, between 2003 and 2004 and between 2008 and 2009.

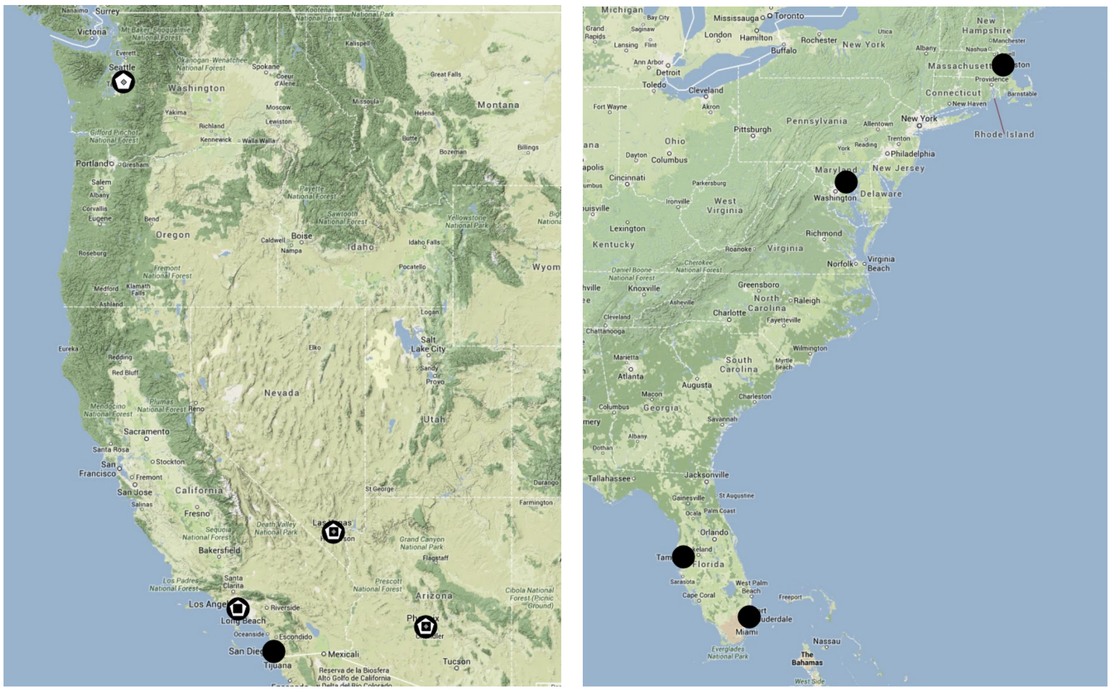


Figure 5.1: Airports from patterns extracted from "Last 20 years" using Jaccard similarity and $\sigma = 0.5$.

We also present a similar experiment using the diameter constraint with $\Delta = 1$ (maximum diameter, p.49), i.e., the pattern is a clique, and a volume threshold $\vartheta = 150$. A clique represents a set of airports connected by pairs by a direct flight, i.e., it will often be either important airports

over the United States or locally connected airports. Volume threshold is set higher as the the airports are highly connected and patterns are larger.
Seven patterns are extracted concerning an overlapping set of airports presented in figure 5.2. All the patterns present an increasing in the attributes “nbDep”,“nbArr”,“nbCan”,“depDel”,“arrDel” at a subset of the time steps 1993-1994, 1994-1995 and 1999-2000.

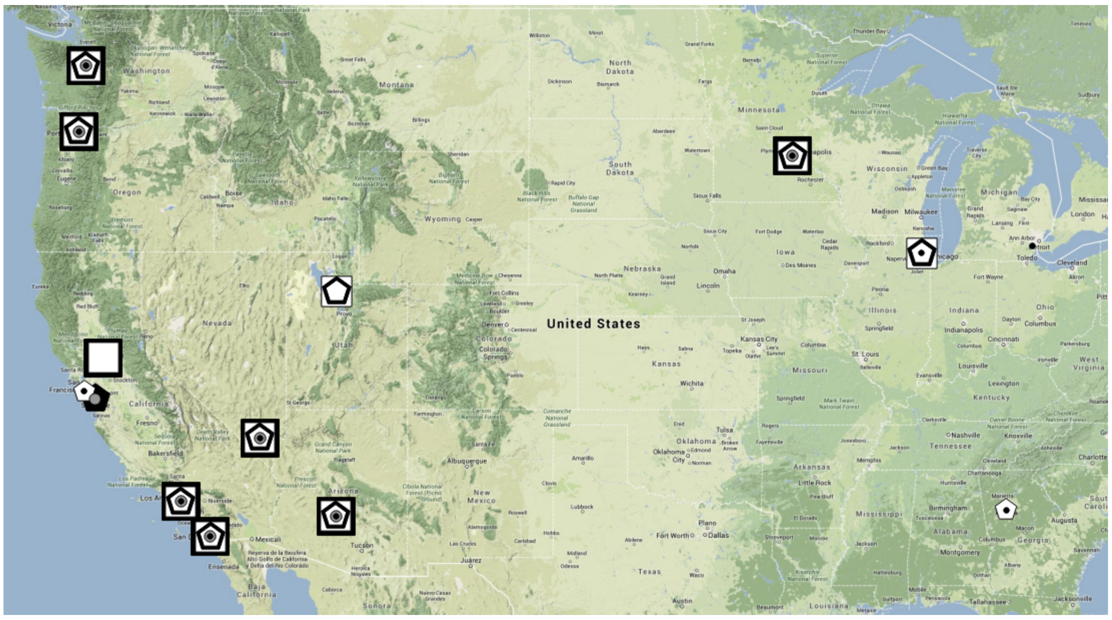


Figure 5.2: Airports from patterns extracted from “Last 20 years” using diameter and $\Delta = 1$.

Considering these extractions, we can conclude that there is too much regularity in the dataset to highlight interesting trends with only these constraints. The patterns are similar using similarity or diameter, however similarity allows a slightly better way to constrain the pattern and to answer specific needs of the expert. Notice that whatever the extraction, executions times have taken less than few minutes.

Two years around '9/11'

For this dataset, the parameters of the experiment were set to $\sigma = 0.6$, $\vartheta = 70$ using Jaccard similarity. Three patterns were extracted in 10 seconds with a small number of vertices and a large number of timestamps, the airports are represented in figure 5.3. Two patterns (left of figure 5.3) contain 3 airports with 2 in common and have a number of departure and arrival flights that increase in October and December 2000, in January, March, May, July, August and December 2001 and in January, March, May, July and August 2002. One pattern (right of figure 5.3) contains 4 airports whose number of departures and arrivals decreased in November 2000, in February, April, June, September, October and November 2001 and in February, June and September 2002.

In all these experiments, we can see that the proposed constraints do not allow to highlight the specific trends of the dataset. However some interesting patterns are extracted concerning the behaviour of groups of airports.

Katrina

We also made an experiment using the Katrina dataset which by definition contains less

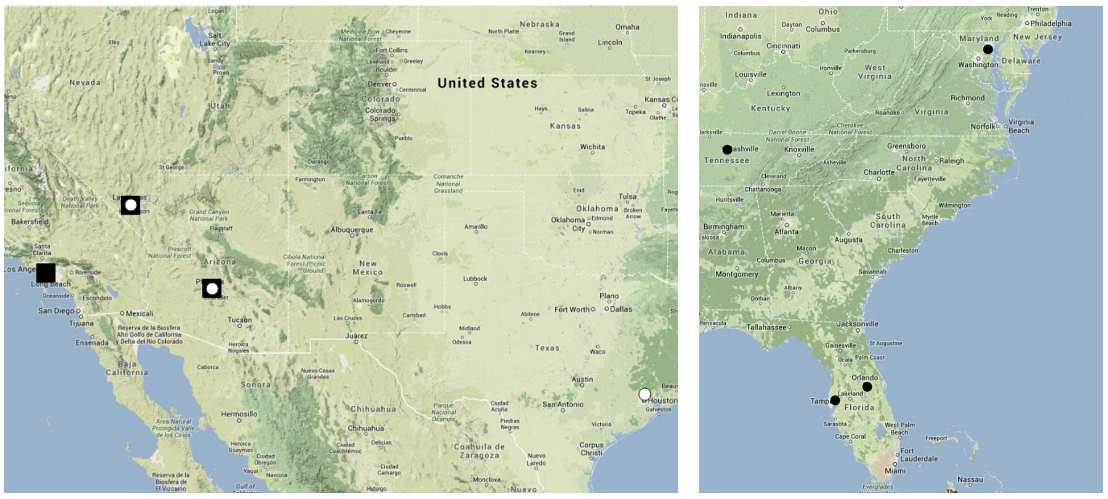


Figure 5.3: Airports from patterns extracted from “Two years around ‘9/11’ ” using Jaccard similarity.

regular trends as it lasts only 8 weeks. To evaluate a new similarity measure, we tried the signature similarity that assesses a similar structural neighbourhood of the vertices. As for the experiment with “DBLP” we set $k_{max} = 5$, $\alpha = 0.3$ (similarity measure parameters, see Section 2.2.2, p.47) and $\sigma = 1$. To avoid redundant trends, we also set $\delta = 0.5$ (minimum evolution, p.47), i.e., attribute values must increase/decrease of at least 50%. 22 patterns were extracted in less than 2 hours and the pattern with the largest volume contains a set of 87 airports whose departure delays and arrival delays decrease between the week before the hurricane and the first week of Katrina. All these airports have between 1 and 6 neighbours, and must be small airports not highly connected.

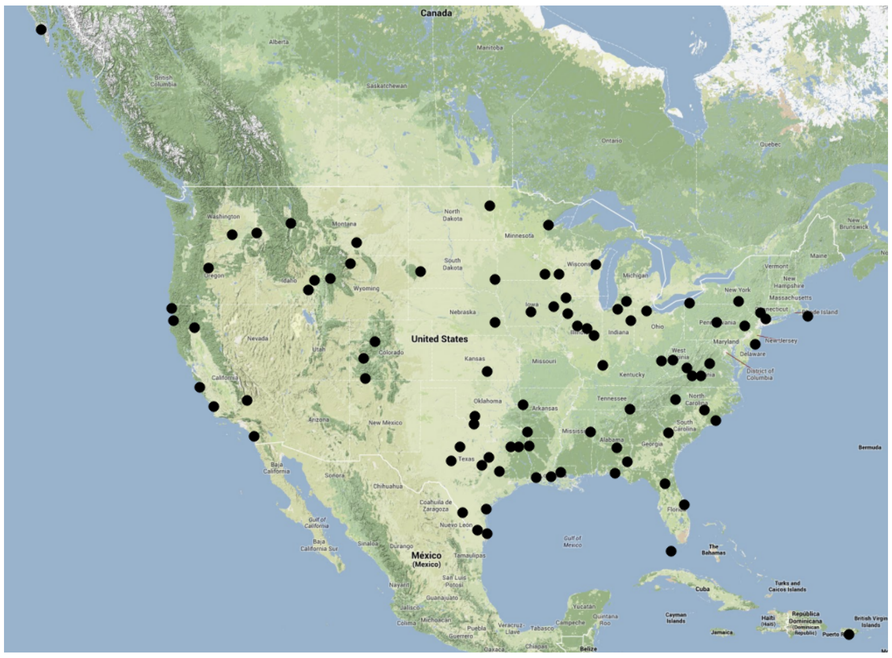


Figure 5.4: Airports from patterns extracted from “Katrina” using signature similarity, $\sigma = 1$ and $\delta = 0.5$.

5.2 Mining Specific Patterns With Regard To Interestingness Measures

Previous experiments show that there is a lot of regularity in the datasets and that use only a trend constraint, a structural constraint and a volume constraint do not allow to extract specific events in the dataset. Let us now present some experiments using outside densities to highlight the patterns that show a specific behaviour considering the rest of the dataset.

Katrina:

To characterize the impact of the hurricane Katrina on the US domestic flights, we set constraints as follows: $\vartheta = 10$ (minimum volume, p.51), $\kappa = 0.6$ (maximum vertex specificity, p.68), $\tau = 0.2$ (maximum temporal dynamic, p.69), $\rho = 0.1$ (minimum trend relevancy, p.71) and $\Delta = |\mathscr{V}| - 1$ (maximum diameter, p.49). We extract 37 patterns in 14 seconds. Let us study two of these patterns: (i) the pattern with the smallest *temporalDynamic* value, and (ii) the pattern with the highest *trendRelevancy* value. These patterns and Katrina's track[2] are shown in Figure 5.6.

Pattern (i) involves 71 airports (in red on Figure 5.6 (left)) whose arrival delays increase over 3 weeks. One week is not related to the hurricane but the two others are the two weeks after Katrina caused severe destruction along the Gulf coast. This pattern has a $temporalDynamic = 0$, which means that arrival delays never increased in these airports during another week. The hurricane strongly influenced the domestic flight organization.
Pattern (ii) has a *trendRelevancy* value equal to 0.81 and includes 5 airports (in yellow on Figure 5.6 (left)) whose number of departures and arrivals increased over the three weeks following Katrina hurricane. Three out of the five airports are in the Katrina's trajectory while the two other ones were impacted because of their connections to airports from damaged areas. Substitutions flights were provided from these airports during this period. The values on the other interestingness measures show that this behaviour is rather rare in the rest of the graph ($vertexSpecificity = 0.29$, $temporalDynamic = 0.2$).

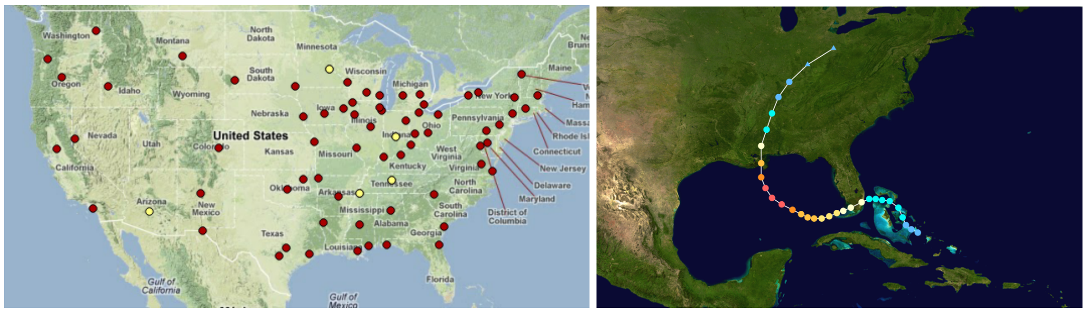


Figure 5.5: Airports (left) involved in the top temporalDynamic pattern (in red) and in the top trendRelevancy (in yellow) and the Katrina's track (right).

September 2001:

To characterize the impact of September 11 attacks, we look for patterns involving many airports (at least 100) whose trends are relevant, i.e., with $min_V = 100$, $\rho = 0.8$ and a maximum diameter $\Delta = |\mathscr{V}| - 1$. The other thresholds are set in such a way they do not constrain the pattern, i.e., $\vartheta = \min_T = min_A = 1$ (minimum number of times and attributes, p.51) and $\kappa = \tau = 1$. Given this setting, the algorithm returns 3 patterns in 8 seconds. These patterns are reported in Table 5.1. They depict a large number of airports, whose number of cancelled flights increased

[2]Map from ©2013 Google, INEGI, Inav/Geosistemas SRL, MapLink
`http://commons.wikimedia.org/wiki/File:Katrina_2005_track.png`

on September 11 and 12 compared to the previous days, and then decreased two days after the terrorist attacks (between the 13th and 16th September). These patterns identify the time required for a return to normal domestic traffic.

Pattern	$\lvert V \rvert$	Days	Ω	*vertexSpec.*	*temporalDyn.*	*trendRel.*
P1	179	10, 11	nbCan$^+$	0.5	0.41	0.94
P2	111	13, 15	nbCan$^-$	0.52	0.83	0.9
P3	102	13, 14, 15	nbCan$^-$	0.6	0.84	0.81

Table 5.1: Trend dynamic sub-graphs extracted by MINTAG on September 2001 graph.

The trend relevancy of these three patterns is high which shows that these sets of airports have really different behaviour on the other attributes. Vertex specificity and temporal dynamic, however, are relatively high. Indeed, we’ve seen earlier that this trend is often followed by the US airports, but the pattern as a whole is really specific to these dates.

Two years around 9/11:

Considering longer periods before and after the September attacks, with more restrictive threshold values, i.e., with $\Delta = |\mathcal{V}| - 1$, $min_V = 100$, $\kappa = 0.5$ and $\rho = 0.8$, we obtain 87 patterns in 67 seconds. The top *trendRelevancy* pattern involves 159 airports that have an increasing number of cancelled flights in September 2001 and December 2000. Obviously, the number of cancelled flights in September 2001 is related to terrorist attack. It is noteworthy that December 2000 snow storm had a similar impact on the cancellation of flights, because we do not quantify the strength of the trends. Actually, the number of cancelled flights in September 2001 is four times bigger than the one in December 2000.

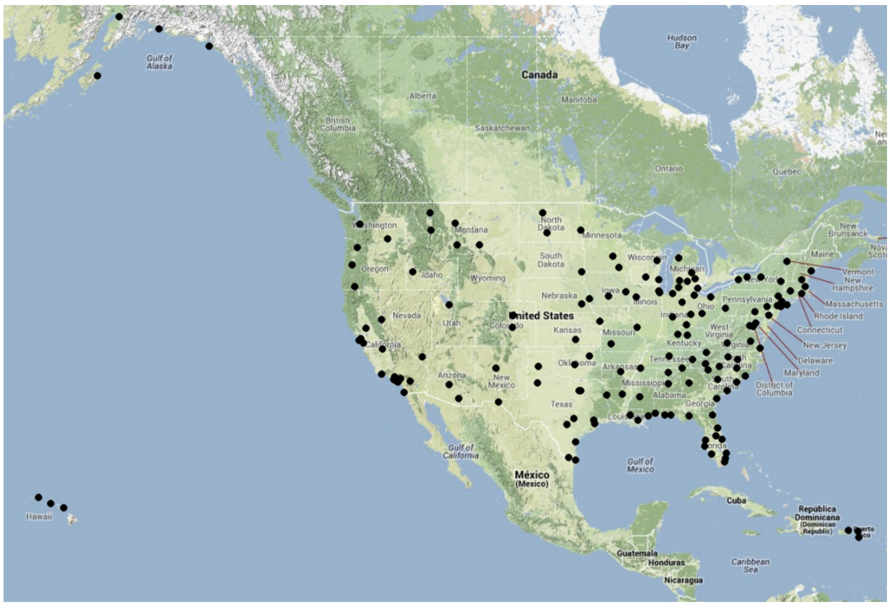


Figure 5.6: Airports involved in the top trendRelevancy pattern .

It is noteworthy that the use of density measures allows to extract patterns that denote more specific event instead of regular schedule changes.

5.3 Mining Hierarchical Co-Evolution Patterns

To evaluate the impact of the hierarchy on this dataset let us make two experiments, one with $\gamma = 1$ and one with $\gamma = 1.2$. γ is the minimum gain of purity (p.81) needed to specialize an attribute, the higher γ, the less attributes are specialized. The aim is to compare the extracted pattern while reducing the possibility of specialization and then while keeping general patterns.

September 2001

With the dataset "September 2001", the aim is to extract patterns that reveal events that affect many airports. Then parameters are set to $min_V = 40$ (minimum number of vertices, p.51), $\kappa = 0.2$ (maximum vertex specificity, p.68), $\tau = 0.4$ (maximum temporal dynamic, p.69), $\psi = 0.9$ (minimum purity, p.80) and $\gamma = 1$ or $\gamma = 1.2$. In both extractions, results are obtain in 2 seconds ; 6 patterns are extracted in the first experiment and 4 in the second one. In figure 5.7 is proposed a comparison of the results of both extractions. Light coloured circles represent the first extraction (i.e., with $\gamma = 1$) and dark coloured circles and bold attributes represent the second extraction (i.e., with $\gamma = 1.2$).

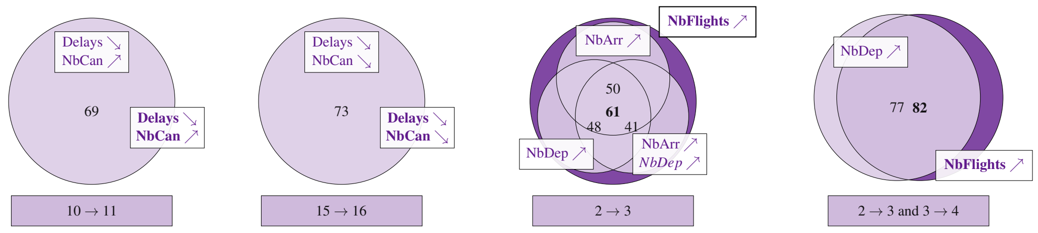


Figure 5.7: Comparison of the resulting patterns with "September 2001" while varying γ. Light coloured circles represent the first extraction (i.e., with $\gamma = 1$) and dark coloured circles and bold attributes represent the second extraction (i.e., with $\gamma = 1.2$).

First there are two patterns that are identical concerning a decreasing in the delays and a decreasing or increasing of the number of cancellation just after "9/11" or later. Indeed, the attribute *nbCan* is much more specific than its parent *nbDisturb* and the behaviour on the attribute *NbDiv* is different, then keeping the parent attribute would bring a false information.

Three patterns of the first extraction are grouped in only one pattern of the second extraction. In the first extraction, the attributes are the number of departures and/or of arrivals that increase. In the second extraction the pattern contains an increasing number of flights which is the "parent" attribute. Concerning the vertices, the totality of the airports of the 3 patterns are contained in the pattern found using $\gamma = 1.2$.

Finally, a pattern is found in the first extraction with 77 airports and an increasing number of departures. In the second extraction, a pattern is found with 82 airports and an increasing number of flights. However, the airports of the second pattern do not contain the totality of the airports of the first extraction, two airports are discarded. Indeed, the purity of the pattern is near the minimum support and the addition of these airports would probably lead to a non valid pattern.

Katrina

For the dataset "Katrina" we want to look for frequent trends. Then parameters are set to $min_V = 50$, $min_T = 3$ (minimum number of times, p.51), $\psi = 0.9$ and $\gamma = 1$ or $\gamma = 1.2$. In both extractions, results are obtain in 1 second ; 12 patterns were extracted in the first experiment and 6 in the second one. In figure 5.8 is proposed a comparison of the results of both extractions. Light coloured circles represent the first extraction (i.e., with $\gamma = 1$) and dark coloured circles and bold attributes represent the second extraction (i.e., with $\gamma = 1.2$).

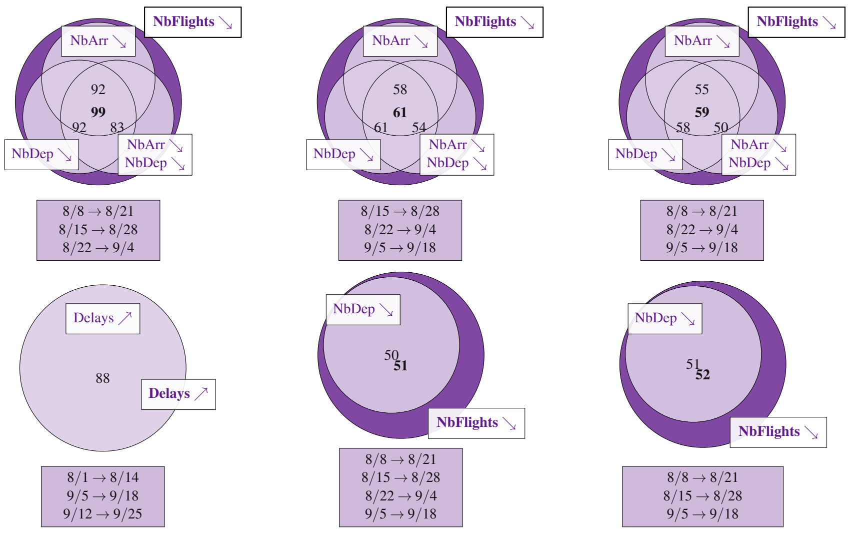


Figure 5.8: Comparison of the resulting patterns with “Katrina” while varying γ. Light coloured circles represent the first extraction (i.e., with $\gamma = 1$) and dark coloured circles and bold attributes represent the second extraction (i.e., with $\gamma = 1.2$).

There are three different behaviours. Some are similar to those presented using “September 2001”. First sets of 3 patterns in the first extraction, are grouped in 1 pattern in the second extraction. This behaviour is followed by three groups presented in the top of Figure 5.8. It is similar with the results of “September 2001”, i.e., the “general” pattern (e.g., on the top left example, the pattern with 99 airports and a decreasing number of flights) contains all the airports of the three specialized patterns (e.g., on the top left example, the patterns with respectively 92,92 and 83 vertices) and the “parent” attribute (e.g., in the top left example “NbFlights” is the parent attribute of “NbDep” and “NbArr”). Second, a pattern is found identical in the two extractions, it is presented in the bottom left example of Figure 5.8 with 88 airports, 3 timestamps and increasing delays.

Two patterns are also found larger in the second extraction, i.e., with the “parent” attribute and more airports. These two examples are presented in the bottom right of Figure 5.8. The first example is a pattern that contains 50 airports that have a decreasing number of departure in the first extraction (i.e., with $\gamma = 1.2$) and 51 airports that have a decreasing number of flights in the second extraction (i.e., with $\gamma = 1.2$) including the 50 vertices of the first extraction. The second example is similar with respectively 51 and 52 airports. It is noticeable that even if the pattern in the first extraction concerns only the number of departures and not the number of arrivals, the patterns extracted in the second experiment with the number of flights have a purity greater than 90%. Then the patterns in the second extraction provide more information.

5.4 Mining Skypatterns

The regularity of trends in this dataset implies a large number of possible patterns. Then, while using skylines, there is a need to fix some of the thresholds and not using some of the measures as dimensions of the skyline. We choose to fix all or part of the size measures. Indeed, we’re interested in events that affect a large part of the airports and forcing the minimum number of timestamps allows to evaluate how these event affect the us domestic flights on a long term.

"September 2001"

First, we choose to fix all the thresholds except for the purity and the density constraints that are the most difficult to evaluate. Figure 5.11 presents the airports in the pattern extracted from "September 2001" using $\Delta = 1$ (the maximal diameter, p.49),$\gamma = 1.2$ (the minimum gain of purity, p.81),$\vartheta = 90$ (the minimum volume, p.51),$min_V = min_T = min_A = 2$ (the minimum number of vertices and times, p.51 and the minimum number of attributes, p.83). One pattern was found which is better than others together on purity, vertex specificity and temporal dynamic.

The airports are all directly connected the 9, 15 and 16 of September 2001 and they have decreasing delays and an increasing number of flights between these days and the day after. This pattern shows the recovery of air traffic in these highly connected airports after "9/11". Moreover, there is also a same evolution between the 9 and the 10 of September 2001 which shows that their traffic seems to be really depending of each other as there was no major event for the United-States at this date. Indeed analysing more in details this result, the purity of the pattern is of 95% which is really high. Indeed, as "9/11" touched all the country, the vertex specificity is relatively high with $specificityV = 0.54$. And as the number of flights change in a planned manner between days and delays can change often, temporal dynamic is also relatively high with $dynamicT = 0.47$.

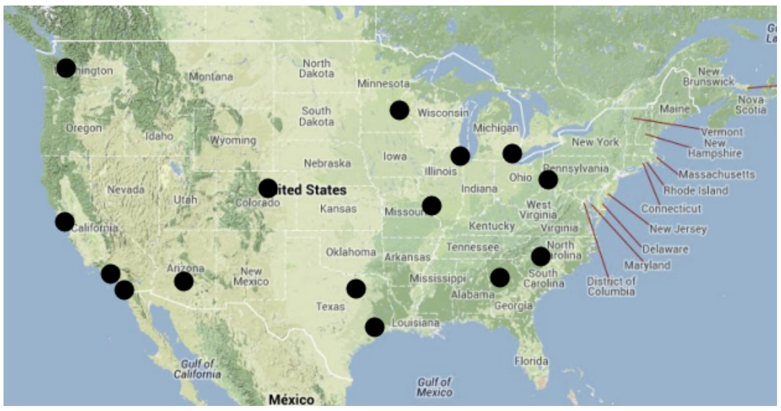


Figure 5.9: Airports in the pattern extracted from "September 2001" using $\Delta = 1, \gamma = 1.2, \vartheta = 90, min_V = min_T = min_A = 2$ and other measures as skyline dimensions.

"Katrina"

Using the Katrina dataset, we want to study the impact of the skylines. Then, we choose to fix the thresholds for all the size measures and for the purity measure in a way to not constrain too much the patterns but enough to extract relevant patterns. Parameters are set to: $\Delta = |\mathscr{V}| - 1$, $\vartheta = 5$, $min_V = 5$, $min_T = min_A = 1$, $\gamma = 1.1$ and $\psi = 0.9$ (the minimum purity, p.80). Both density measures are used as dimensions of the skyline.

The algorithm finds 8 patterns on the skyline, these patterns are presented in Table 5.2 and in Figure 5.10 with regard to their value of vertex specificity and temporal dynamic with coloured dots linked together. Let us recall that the two measures must be minimized, then the best patterns are the one with the minimal values on both measures, that's why the skyline is drawn up to the dots. We then made an experiment without skyline and using the highest values of the skypatterns for specificityV and dynamicT thresholds, i.e., $\kappa = 0.4$ and $\tau = 0.61$. The algorithm then finds 53 valid patterns presented with black dots in figure 5.10. We can conclude that the skyline allows to avoid the extraction of at least 45 patterns that seems to be less interesting.

	$\|V\|$	T	A	purity	VS	TD
P_1	213	08/22 → 09/4	nbFlights$^-$	0.96	0	0.61
P_2	31	09/5 → 09/17	nbDiv$^-$	1	0.004	0.39
P_3	39	08/22 → 09/04	nbDiv$^-$	1	0.008	0.36
P_4	10	08/8 → 08/21 ; 08/22 → 09/04	nbDiv$^-$	1	0.15	0.3
P_5	57	08/29 → 09/17	nbFlights$^+$	0.92	0.25	0.27
P_6	9	08/29 → 09/24	nbArr$^+$	1	0.288	0.084
P_7	5	08/29 → 09/24	nbDep$^+$, nbArr$^+$	1	0.29	0.075
P_8	8	08/8 → 08/28 ; 08/29 → 09/11 ; 09/12 → 09/24	nbCan$^-$	1	0.39	0

Table 5.2: Skypatterns of the extraction on the Katrina dataset (VS = vertexSpecificity, TD = temporalDynamic).

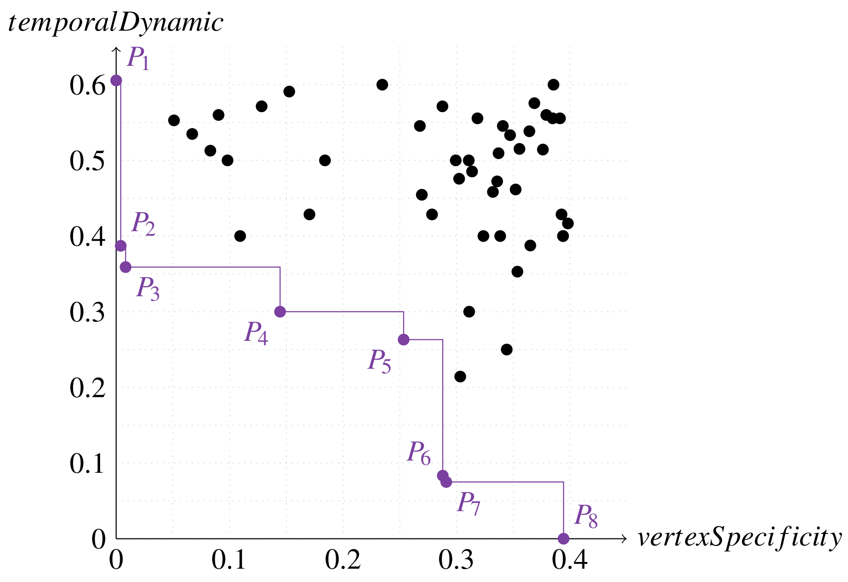


Figure 5.10: Plot of the extracted patterns w.r.t. their value of vertex specificity and temporal dynamic. Patterns of the skyline are coloured and linked together.

Moreover, to extract only these 53 patterns, we need to know the best values for κ and τ. With higher values of threshold (i.e., until $\kappa = \tau = 1$) the algorithm finds up to 218 patterns.

Let us study the two patterns at both extremities of the skyline, i.e., P_1 and P_8. First the pattern P_1, with $vertexSpecificity = 0$ and $temporalDynamic = 0.6$, the airports are presented in figure 5.11 on the left. This pattern contains 213 airports whose number of flights decreased between '08/22' and '09/4', i.e., the number of flights decreased during the hurricane. This pattern contains 76% of the airports of the dataset and it also contains the totality of the airports whose number of flights decrease. Moreover, the purity of the pattern is of 95%. However, if this decreasing must be partly caused by the hurricane it must also be caused by the end of the summer holidays.

Airports of the pattern P_8, with $vertexSpecificity = 0.39$ and $temporalDynamic = 0$ are presented in figure 5.11 on the right. This pattern contains 8 airports whose number of cancelled flights decreased between '8/8' and '8/28', between '8/29' and '9/11' and between '9/12' and '9/25', i.e., during half the weeks of the dataset but not continuously. As purity is obviously

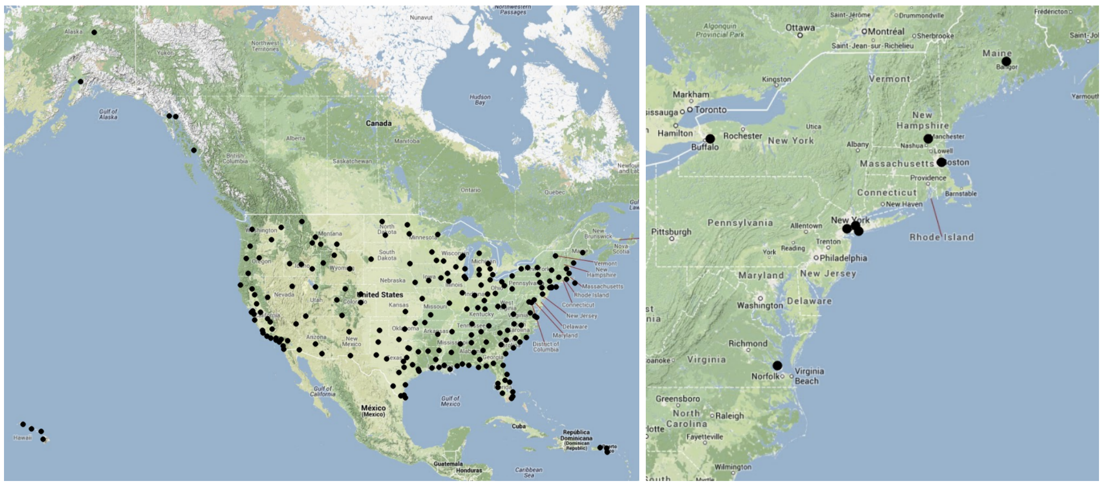


Figure 5.11: Airports in the patterns extracted from “Katrina” using $\Delta = |\mathcal{V}| - 1, \gamma = 1.1, \vartheta = 5, min_V = 5$, $min_T = min_A = 1$, $\psi = 0.9$ and density measures as skyline dimensions.

equal to 1 and $temporalDynamic = 0$, this pattern tends to prove that these airports are highly dependant on each others. Moreover, the vertex specificity is not that high which means that the behaviour of the pattern is not that current.

5.5 Discussion On The Results

The dataset created using RITA “On-Time Performance” database contains a lot of regular trends due to week-end, seasons, holidays, etc. Then, the use of only trend constraint, structural constraint, and size constraints does not allow to highlight the patterns of real interest as there are too many patterns extracted that depict regular schedules of the datasets. However, since the Katrina dataset contains less regularities than the other datasets, some interesting patterns can be found.

The use of densities allows to exclude a large part of non-interesting patterns. Using the proposed datasets, the aim of extracting patterns that characterize the events of ’9/11’ and Katrina is then reached. In parallel the use of a hierarchy, even if it is a quite simple one, allows to reduce the number of patterns by resuming them. If the information found can be slightly different, most of the information is not lost and even more is given.

Finally, the use of skyline analysis on two of the datasets allows not only to extract interesting patterns, but also to reduce their number by only keeping the best ones, facilitating the interaction with the non data-miner user. Nevertheless, many measures have to be fixed and left outside of the skyline in order to not find too many patterns. Indeed, using the parameters on the previous section on the Katrina dataset, we made experiments with one to six measures in the skyline. We obtained a mean of 4 patterns using either one or two measures to a mean of 51 patterns with six measures. With all the measures used in the skyline analysis we obtained 168 patterns.

6 — Conclusion

Graphs have been widely studied in the data-mining community, however, it is not the case of dynamic attributed graphs. Yet, this type of graphs provides many ways to represent real-world phenomena and many possibilities of studying it and discovering information. In this thesis we propose to extract local co-evolution patterns that describe specific trends among some subgraphs.

Summary of our contributions

The core of this work is the definition of co-evolution patterns, i.e., trisets of vertices, timestamps and signed attributes. These patterns aim at describing subgraphs of the dynamic attributed graph that follow specific behaviours. To this end, they must satisfy two generic constraints, a constraint related to the graph structure and a constraint related to the co-evolution of the attribute values.

We propose two possible constraints relying on the graph structure, i.e., the cohesiveness and the diameter.

- Cohesiveness: to make sure that the pairs of vertices are similar, i.e., that the neighbourhood of each pair of vertices is similar. It is based on a predicate which returns true if vertices are considered as similar and false otherwise, i.e., if the similarity measure is greater than a given threshold σ. Any similarity measure can be used and we propose the use of four of them: cosine, Jaccard, activation and signature similarities. A pattern respects the constraint if the predicate is true for each pair of vertices at each timestamp.
- Diameter: to make sure that the diameter of the subgraph induced by the vertices of the pattern is lower than a given threshold Δ at each timestamp. The threshold provides information on the type of pattern. A threshold $\Delta = 1$ implies a clique on the vertices of the pattern. A threshold $\Delta = 2$ implies that each pair of vertices has at least one neighbour in common in the subgraph induced by the vertices of the pattern, which includes quasi-cliques. Then the higher the threshold, the sparser the induced subgraph until a threshold Δ equal to the number of vertices of the graph minus one that implies only that the subgraph is connected.

Empirical studies made on different datasets indicate that even if the use of similarity measures can allow to extract relevant patterns, the use of a diameter constraint is often more interesting.

Indeed similar patterns are extracted with diameter in a shorter execution time and the threshold is easier to fix for the user. Moreover, due to the non-transitivity of the similarity measure, the cohesiveness constraint tends to fragment the patterns instead of the diameter constraint.

To evaluate the co-evolution of the attribute values, we first define a strict evolution on the set of attributes of the dataset, i.e., a decreasing or an increasing of the attribute value. The constraint makes sure that the attribute values of each vertex follow the (decreasing/increasing) trend associated to the attribute between each timestamp of the pattern and its consecutive timestamp. We offer the possibility to use a threshold δ that constrain the value of the attribute at time $t+1$ to be higher (resp. lower) than $1+\delta$ (resp. $1-\delta$) its value at time t. To improve the quality of the patterns and avoid the extraction of several redundant patterns we then propose to use additional user-knowledge with the dataset. This additional information is a hierarchy on the set of attributes, i.e., new attributes "parent" of the dataset attributes are defined. A new constraint of evolution must then be proposed as the "parent" does not have any value associated and no direct trend can be computed. We define the hierarchical strict evolution that computes a value for each "parent" attribute by adding the value of its "children" attributes. Once this value is associated to the "parent" attributes, as for strict evolution, the constraint makes sure that the attribute values of each vertex follow the (decreasing/increasing) trend associated to the attribute between each timestamp of the pattern and its consecutive timestamp. We also offer the possibility to use a threshold δ with the constraint of hierarchical strict evolution.

The use of such a hierarchy on the attributes allows to obtain generalized patterns, i.e., patterns that provide information not on every specific attributes but also on general attributes that describe multiple specific attributes. It implies two main questions: Does the trend followed by the "parent" attribute reflects those of its "children"? And which information is the more relevant between the "parent" and the "children" attribute? We define two measures to answer these questions. A measure of purity that evaluates the proportion of "children" attributes that follow the trend associated to their "parent". If this proportion is greater than a threshold ψ, the "parent" attribute is considered as representative. To choose between different level of hierarchy, we propose a measure of gain of purity that evaluates the purity of the pattern with the "children" attributes compared to the purity of the same pattern with the "parent" attributes. If the purity of the pattern with the "children" attributes is greater of more than a given threshold γ, the pattern is considered as more relevant.

To extract patterns more specific with regard to the rest of the dynamic attributed graph, we also propose density measures. These measures aim at revealing the pattern that are specific with regard to either other vertices, other timestamps or other attributes, i.e., they enable to answer these three questions: How similar are the vertices outside the co-evolution pattern to the ones inside it? Does the pattern appear suddenly at the timestamps of the pattern or does-it also appear at other times? Does the vertices of the pattern have a similar behaviour only on the attributes of the pattern or also on the attributes outside? We define three measures, one on each set (vertices,times,attributes). The vertex specificity computes the proportion of vertices outside the pattern that follow the behaviour of the pattern, if this proportion is lower than a threshold κ, the pattern is considered specific. The temporal dynamic computes at each timestamp outside the pattern how much the behaviour of the pattern is respected, if the proportion of trends respected at each timestamp is lower than a threshold τ, the pattern is considered specific. Finally, the trend relevancy evaluates if the similarity of behaviour of the elements in the pattern with attributes outside the pattern with a measure of entropy, if this entropy is greater than a threshold ρ, the pattern is considered specific. This last measure is not applicable to the use of a hierarchy on the attributes, indeed, it contradicts with the concept of gain of purity. We also use current constraints to obtain more relevant pattern, size constraints and maximality.

We propose two constraint-based algorithms (MINTAG and H-MINTAG) to extract both co-evolution patterns and hierarchical co-evolution patterns. These algorithms use the anti-monotonic and piecewise anti-monotonic properties of the constraints to reduce the search space and extract patterns in a reasonable execution time. However the use of these algorithms imply to fix many thresholds, the threshold boundary may be arbitrary and it is not obvious that patterns which respect a subset of them are not relevant. That's why we propose to extract skypatterns among a multidimensional space as a chosen subset of the defined measures. The use of a skyline simplifies the use of the algorithm by the user. A chosen subset of the defined measures is used as dimensions of the skyline while the other measures are used as constraints with a defined threshold. This allows to reduce the number of threshold to fix, avoid to arbitrarily set some of them and patterns that are relevant considering only a subset of the measures can be extracted. Moreover, the use of a skyline enables to highlight the most relevant patterns with regard to the measures.We then design a third algorithm to extract the skypatterns (Sky-H-MINTAG).

To evaluate our methods, we present a quantitative and qualitative empirical study on several real-world dynamic attributed graphs, especially spatio-temporal datasets. We show that the methods allow to extract relevant patterns and that the algorithms run in a feasible execution time.

Perspectives

The findings suggest the following directions: future work should concentrate on both scalability issues and the semantics conveyed by the patterns, in particular with regard to user prior knowledge.

Enhancing the computation of the diameter.

The computation of the diameter is a limit of our algorithm. The diameter of the induced subgraph is computed nearly at each step of the algorithm and it is time consuming. We observe in the experiments that it is only tractable when the diameter threshold is low.
A way of improving the execution times could be to compute a lower bound of the diameter instead of the real diameter during the execution. If some pruning would be less strict, it should be compensated by the gain in computation time. To this aim, we could take benefit of some properties of the diameter, such as the ones presented in [39] and use the randomized BFS (Breadth First Search) procedure that uses two consecutive BFS to compute a lower bound of the diameter of the graph.

Parallel computing to scale on large amount of data.

Parallel execution of an algorithm on multi-core architecture enables to improve the performance [70]. The enumeration strategy of our algorithm enables a straightforward parallelization. The original search space is decomposed into portions that can be independently computed in main memory, these portions can also be computed separately with a parallelized code. At each node of the enumeration, one instance of the parallelized code could treat each of the branches. For instance, as the first step of the algorithm is to insert a vertex v from C into P, the computation of all possible co-evolution patterns from each node $P = \{(v), \varnothing, \varnothing\}, C = \{(\mathscr{V} \setminus v), (\mathscr{T}), (\mathscr{A} \times \{-,+\})\}, D = \{\varnothing, \varnothing, \varnothing\}$ could be parallelized for each $v \in \mathscr{V}$.

Exploiting user prior knowledge.

We have developed a preliminary work on the use of additional user-knowledge by proposing the use of a specific type of hierarchy on the attributes where each attribute has at most one parent. However other ways of representing user-knowledge could be imagined.
First, a more complex hierarchy with multiple parents could be interesting. For instance with the “DBLP” dataset, a journal can be associated both to “databases” and to “data mining”, the hierarchy would then be more accurate. Other hierarchies on the timestamps or on the vertices can also be considered. For instance with “Domestic US Flights” datasets, a hierarchy on timestamps could be to generalize by day, week, month, etc.. A hierarchy on the vertices could be lead by user knowledge, as for instance US states for the airports, or could be lead, for instance, by computed communities.

Softness in skypatterns analysis.

The use of a skyline allows to avoid thresholds as it highlights the best patterns considering a set of measures. However, the extraction of skypatterns answers to a strict framework and only patterns on the skyline are extracted. However, some relevant patterns could also be found near the skyline. For instance, in our experiments with the Katrina dataset, some patterns extracted using the highest values of the skypatterns as thresholds also seem to stand considering the two measures of interestingness but are hidden by the skyline. Such patterns can be really relevant for the expert to analyse, especially if he is not interested by the extracted skypatterns. The idea would be to offer the possibility to study the patterns behind the skyline. Such a possibility have already been investigated in [88] where the authors propose the extraction of soft-skypatterns in transaction databases. It would be interesting to revisit such skypatterns in the context of the extraction of co-evolution patterns in dynamic attributed graphs.

Appendices

A — Dataset Description

Main characteristics of the datasets are presented in Table A.1.

Dataset		$\|\mathscr{V}\|$	$\|\mathscr{T}\|$	$\|\mathscr{A}\|$	density
DBLP		$2,145$	10	43	1.3×10^{-3}
Brazil landslides		$394,885$	2	11	5.7×10^{-4}
US Flights	Last 20 years	361	20	8	3.2×10^{-2}
	Two years around 9/11	234	25	8	5.7×10^{-2}
	September 2001	220	30	6	5.7×10^{-2}
	Katrina	280	8	8	5×10^{-2}

Table A.1: Main characteristics of the dynamic attributed graphs.

A.1 DBLP

The graph is build from a subset of the digital library “DBLP” which is a computer science bibliography (`http://dblp.uni-trier.de/`). Vertices of the graph represent 2,145 authors who published at least 10 papers in a selection of 43 conferences and journals of the Data Mining and Database communities (see Table A.2) between January 1990 and December 2012. This time period is divided in 10 timestamps, for sake of consistency in the data, we created overlapping periods. Each time stamp describes the co-authorship relations and the publication records of the authors over 5 consecutive years, and two consecutive periods have a 3 year overlap ([1990-1994][1992-1996]...[2008-2012]). Each edge at a time stamp t links two authors who co-authored at least one paper in this time interval. Each vertex at each time is associated to a set of 43 attribute values corresponding to the number of publications in the 43 selected conferences and journals during the related period. To summarize, this dataset consists in 2,145 vertices, 10 timestamps and 43 attributes. This dataset is singular as there is a large scale of conferences/journals in which each author will have a significant set with no publication all along their carrier.

Structural characteristics of the dataset are presented in Table A.3. The structure of the graph evolves significantly between 1990 and 2012. Indeed, all the authors present in the dataset did

Nb	Short Name	Name
0	DMKD	Data Mining and Knowledge Discovery
1	SAC	ACM Symposium on Applied Computing
2	ICDM	IEEE International Conference on Data Mining
3	CommunACM	Communications of the ACM
4	IEEE-TKDE	IEEE Transactions on Knowledge and Data Engineering
5	IEEEIntSys	IEEE Intelligent Systems
6	SIGKDDExp	SIGKDD Explorations
7	DASFAA	Database Systems for Advanced Applications
8	IJCAI	International Joint Conference on Artificial Intelligence
9	IDA	Intelligent Data Analysis
10	ICDE	International Conference on Data Engineering
11	PAKDD	Pacific-Asia Conference on Knowledge Discovery and Data Mining
12	AAAI	AAAI Conference on Artificial Intelligence
13	PVLDB	Proceedings of the VLDB Endowment
14	CIKM	International Conference on Information and Knowledge Management
15	ECML-PKDD	European Conference on Principles of Data Mining and Knowledge Discovery
16	KDD	Knowledge Discovery and Data Mining
17	VLDB	Very Large Data Bases Conference
18	EDBT	International Conference on Extending Database Technology
19	IntellDtAnal	Intelligent Data Analysis (Journal)
20	ICML	International Conference on Machine Learning
21	ICDT	International Conference on Database Theory
22	SDM	SIAM International Conference on Data Mining
23	KnowlInfSyst	Knowledge and Information Systems
24	TKDD	ACM Transactions on Knowledge Discovery from Data
25	PODS	Symposium on Principles of Database Systems
26	VLDBJ	The VLDB Journal
27	StatAnalDtMining	Statistical Analysis and Data Mining
28	SIGMOD	ACM SIGMOD Conference
29	MachineLearning	Machine Learning
30	ACMTransDBSys	ACM Transactions on Database Systems
31	JMLR	Journal of Machine Learning Research
32	ACMTransInfSys	ACM Transactions on Information Systems
33	JIntellInfSys	Journal of Intelligent Information Systems
34	InfSys	Information Systems
35	DataKnlEng	Data and Knowledge Engineering
36	PatternRecog	Pattern Recognition
37	BioInfo	Bioinformatics
38	BMCBio	BMC Bioinformatics
39	ECAI	European Conference on Artificial Intelligence
40	WWW	International World Wide Web Conferences
41	DEXA	Database and Expert Systems Applications
42	ILP	International Workshop on Inductive Logic Programming

Table A.2: Conferences and Journals selection of “DBLP” dataset.

not published at each year, mainly in the beginning years. As a vertex must be present at each time they are not connected to any other at certain timestamps. This reflects in the density, the diameter and the number of connected components. These vertices have also a zero value on all their attributes at these timestamps. Added to the fact that authors do not publish at all the conferences/journals, the dataset is constituted of 95% of zero in its attribute values.

The hierarchy A.1 created on the dataset is one possibility among others. As certain conferences or journals are at the border between domains, they are difficult to place and we take the major domain. It they are general, they are higher in the hierarchy.

A.2 Brazilian landslides

This dataset is derived from 2 satellite images taken before and after huge landslides, images are presented in Figure A.2.

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
Density ($\times 10^{-3}$)	0.32	0.43	0.58	0.74	1	1.5	2	2.4	2.5
Diameter	19	15	22	18	18	18	19	20	19
Number of connected components	1708	1599	1426	1265	997	720	490	394	404
% vertex alone	75	70	62	54	42	29	19	15	16

Table A.3: Characteristics of the "DBLP" dataset.

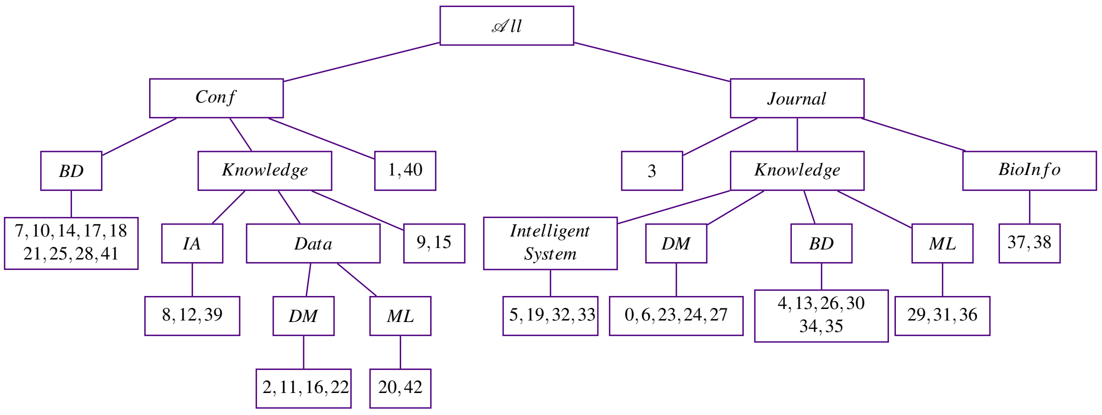


Figure A.1: Hierarchy of the DBLP dataset

This dynamic attributed graph is composed of 394,885 vertices that stand for image shapes (segmented areas) extracted by a segmentation performed in eCognition 8.64 with a scale factor of 20 [7]. There is an edge between two vertices if the corresponding shapes are contiguous. Each segment is associated to 10 attributes that are the spectral response in infra-red (NIR), red (R), blue (B), green G) and indices computed from these values, $\frac{G}{R}$, $\frac{B}{G}$, $\frac{R}{NIR}$, NDVI (i.e., $\frac{NIR-R}{NIR+R}$), red index (i.e., $\frac{R^2}{B\times G^3}$) and brilliance index (i.e., $\sqrt{NIR^2+R^2}$). To summarize this dataset consists in 394,885 vertices, 2 timestamps and 11 attributes. The dataset has a density of 1.46×10^{-3} and is composed of one only connected component.

A.3 Domestic US Flights

The RITA "On-Time Performance" database[1] contains on-time arrival data for non-stop US domestic flights by major air carriers and provides additional items such as departure and arrival delays, origin and destination airports, flight numbers, scheduled and actual departure and arrival times, cancelled or diverted flights, taxi-out and taxi-in times, air time, and non-stop distance.

From this database, we generated 4 dynamic attributed graphs that aggregate data over different period of time. Graph vertices stand for US airports and are connected by an edge if there is at least a flight between the two corresponding airports during the considered time period.
Last 20 years: Data are aggregated over each year, which leads to a sequence of 20 static graphs with 361 vertices.
Two years around 9/11: This dynamic attributed graph aims at studying the impact of 9/11 on the US airports. To this end, airport activities are aggregated over each month between September

[1]http://www.transtats.bts.gov

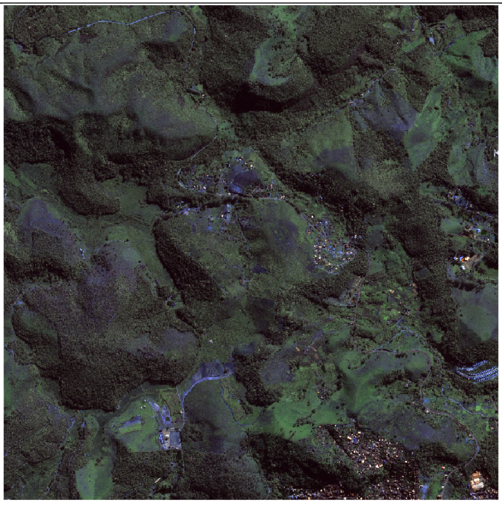
Before landslide	After landslide
	

Figure A.2: Images of Brazil at the 2 timestamps.

2000 and September 2002, leading to 25 static graphs of 234 vertices.
September 2001: Data are aggregated over each day of September 2001, leading to 30 static graphs that involve 220 vertices.
Katrina: To study the consequences of hurricane Katrina on US airports, data are aggregated over each week between 08/01/2005 and 09/25/2005. The resulting dynamic graph has 8 timestamps and involves 280 vertices.

We consider 8 vertex attributes that are:

- **NbDep**: The number of departures from the given airport.
- **NbArr**: The number of arrivals to the given airport.
- **NbCan**: The number of cancelled flights which should have started from the given airport.
- **NbDiv**: The number of flights which have been diverted and which should have arrived to the given airport.
- **depDelay**: The mean delays of departure at the given airport.
- **arrDelay**: The mean delays of arrival at the given airport.
- **taxiIn**: The mean time elapsed between wheels down and arrival at the given airport gate.
- **taxiOut**: The mean time elapsed between departure from the given airport gate and wheels off.

The dataset “September 2001” contains only the 6 first attributes when other datasets contain all of them.

Structural characteristics of the dataset are presented in Table A.4. For each dataset we present the characteristic at the first time, the first third time, the second third time and the last time. Let us study the constraints:

- Density: The density depends on the dataset. However, whatever the dataset, it is relatively dense compare to “DBLP” or “Brazil” and if the density change through time it stays in a same range of values.
- Diameter: We can notice that whatever the dataset and the timestamp, the diameter is small, i.e., equal to 4 or 5. Then using the constraint of diameter, the execution times will be quickly important, and the use of $\Delta = 5$ is equivalent to the use of $\Delta = |\mathcal{V}|$. To obtain interesting patterns in a reasonable running time, the best solution is then to fix $\Delta = 1$ or

Dataset	Characteristic	t_1	$t_{\frac{\lvert\mathscr{V}\rvert}{3}}$	$t_{\frac{2\times\lvert\mathscr{V}\rvert}{3}}$	$t_{\lvert\mathscr{V}\rvert}$
Last 20 years	Density	0.029	0.03	0.039	0.036
	Diameter	5	5	5	4
Two years around 9/11	Density	0.056	0.059	0.057	0.054
	Diameter	5	4	5	5
September 2001	Density	0.065	0.065	0.062	0.061
	Diameter	4	4	4	4
Katrina	Density	0.052	0.052	0.051	0.05
	Diameter	5	5	5	5

Table A.4: Characteristics of the “US Flight” datasets.

$\Delta = \mathscr{V}$.

The hierarchy A.3 was created for this dataset, attributes are grouped by pairs whose semantic similarity is obvious.

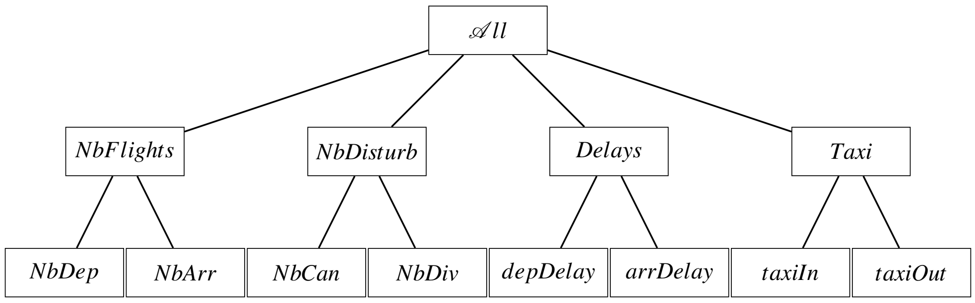


Figure A.3: Hierarchy of the “US Flights” datasets.

7 — Bibliography

[1] James Abello, Mauricio G. C. Resende, and Sandra Sudarsky. “Massive Quasi-Clique Detection”. In: *LATIN*. 2002, pages 598–612 (cited on page 25).

[2] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. “CFinder: locating cliques and overlapping modules in biological networks”. In: *Bioinformatics* 22.8 (2006), pages 1021–1023 (cited on page 25).

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. “Mining Association Rules between Sets of Items in Large Databases”. In: *SIGMOD Conference*. 1993, pages 207–216 (cited on page 16).

[4] Rezwan Ahmed and George Karypis. “Algorithms for Mining the Evolution of Conserved Relational States in Dynamic Networks”. In: *ICDM*. 2011, pages 1–10 (cited on page 36).

[5] Rezwan Ahmed and George Karypis. *Algorithms for Mining the Coevolving Relational Motifs in Dynamic Networks*. Technical report. 2014 (cited on page 36).

[6] Annalisa Appice, Anna Ciampi, and Donato Malerba. “Summarizing numeric spatial data streams by trend cluster discovery”. In: *Data Mining and Knowledge Discovery* (2013), pages 1–53 (cited on page 43).

[7] Martin Baatz. “Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation”. In: *AGIT*. Volume 58. 3-4. Taylor & Francis, Inc., 2000, pages 12–23 (cited on pages 106, 133).

[8] Michele Berlingerio, Francesco Bonchi, Björn Bringmann, and Aristides Gionis. “Mining Graph Evolution Rules”. In: *ECML/PKDD (1)*. 2009, pages 115–130 (cited on pages 16, 17, 23, 34).

[9] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. “As time goes by: Discovering eras in evolving social networks”. In: *PAKDD*. 2010, pages 81–90 (cited on page 17).

[10] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. “Foundations of Multidimensional Network Analysis”. In: *ASONAM*. 2011, pages 485–489 (cited on page 23).

[11] Brigitte Boden, Stephan Günnemann, and Thomas Seidl. “Tracing clusters in evolving graphs with node attributes”. In: *CIKM*. 2012, pages 2331–2334 (cited on page 37).

[12] Petko Bogdanov, Ben Baumer, Prithwish Basu, Amotz Bar-Noy, and Ambuj K. Singh. "As Strong as the Weakest Link: Mining Diverse Cliques in Weighted Graphs". In: *ECML/PKDD (1)*. 2013, pages 525–540 (cited on page 25).

[13] Petko Bogdanov, Misael Mongiovì, and Ambuj K. Singh. "Mining Heavy Subgraphs in Time-Evolving Networks". In: *ICDM*. 2011, pages 81–90 (cited on page 35).

[14] Monica Borda. *Fundamentals in information theory and coding*. Volume 6. Springer, 2011 (cited on page 71).

[15] Karsten M. Borgwardt, Hans-Peter Kriegel, and Peter Wackersreuther. "Pattern Mining in Frequent Dynamic Subgraphs". In: *ICDM*. 2006, pages 818–822 (cited on pages 16, 23, 32).

[16] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. "The Skyline Operator". In: *ICDE*. 2001, pages 421–430 (cited on pages 18, 92).

[17] Björn Bringmann and Siegfried Nijssen. "What Is Frequent in a Single Graph?" In: *Pacific-Asia Conf. on Knowl. Discov. and Data Mining (PAKDD)*. 2008, pages 858–863 (cited on pages 16, 25, 34).

[18] Ali Cakmak and Gultekin Özsoyoglu. "Taxonomy-superimposed graph mining". In: *EDBT*. 2008, pages 217–228 (cited on page 77).

[19] Toon Calders, Bart Goethals, and Szymon Jaroszewicz. "Mining rank-correlated sets of numerical attributes". In: *KDD*. 2006, pages 96–105 (cited on page 43).

[20] Toon Calders, Jan Ramon, and Dries Van Dyck. "Anti-monotonic Overlap-Graph Support Measures". In: *Int. Conf. on Data Mining (ICDM)*. 2008, pages 73–82 (cited on pages 16, 25, 34).

[21] Loïc Cerf. "Constraint-Based Mining of Closed Patterns in Noisy n-ary Relations". PhD Thesis in Computer science. INSA-Lyon, July 2010 (cited on page 16).

[22] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. "Data Peeler: Contraint-Based Closed Pattern Mining in n-ary Relations". In: *SDM*. 2008, pages 37–48 (cited on page 35).

[23] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. "Closed patterns meet n-ary relations". In: *TKDD* 3.1 (2009) (cited on pages 52–54).

[24] Loïc Cerf, Tran Bao Nhan Nguyen, and Jean-François Boulicaut. "Discovering Relevant Cross-Graph Cliques in Dynamic Networks". In: *ISMIS*. 2009, pages 513–522 (cited on page 34).

[25] Deepayan Chakrabarti and Christos Faloutsos. "Graph mining: Laws, generators, and algorithms". In: *ACM Computing Surveys (CSUR)* 38.1 (2006), page 2 (cited on pages 15, 23).

[26] Diane J Cook and Lawrence B Holder. *Mining graph data*. John Wiley & Sons, 2006 (cited on page 15).

[27] Elise Desmier, Marc Plantevit, and Jean-François Boulicaut. "Granularité des motifs de co-variation dans des graphes attribués dynamiques". In: *EGC*. 2014, pages 431–442 (cited on pages 18, 79).

[28] Elise Desmier, Marc Plantevit, Céline Robardet, and Jean-François Boulicaut. "Cohesive Co-evolution Patterns in Dynamic Attributed Graphs". In: *Discovery Science*. Springer, 2012, pages 110–124 (cited on pages 18, 60).

[29] Elise Desmier, Marc Plantevit, Céline Robardet, and Jean-François Boulicaut. “Trend Mining in Dynamic Attributed Graphs”. In: *ECML/PKDD (1)*. 2013, pages 654–669 (cited on pages 18, 75).

[30] Elise Desmier, Marc Plantevit, Céline Robardet, and Jean-François Boulicaut. *Discovery of Skylines for Generalized Co-evolution Patterns in Dynamic Attributed Graphs*. Technical report. 2014 (cited on page 18).

[31] Fabien Diot, Élisa Fromont, Baptiste Jeudy, Emmanuel Marilly, and Olivier Martinot. “Graph Mining for Object Tracking in Videos”. In: *ECML/PKDD (1)*. 2012, pages 394–409 (cited on page 36).

[32] Trong Dinh Thac Do, Anne Laurent, and Alexandre Termier. “PGLCM: Efficient Parallel Mining of Closed Frequent Gradual Itemsets”. In: *ICDM*. 2010, pages 138–147 (cited on page 43).

[33] Martin Ester, Rong Ge, Byron J Gao, Zengjian Hu, and Boaz Ben-moshe. “Joint Cluster Analysis of Attribute Data and Relationship Data”. In: *SIAM SDM*. 2006, pages 246–257 (cited on page 15).

[34] Mutsumi Fukuzaki, Mio Seki, Hisashi Kashima, and Jun Sese. “Finding Itemset-Sharing Patterns in a Large Itemset-Associated Graph”. In: *PAKDD (2)*. 2010, pages 147–159 (cited on pages 28–30).

[35] Wei Gao, Kam-Fai Wong, Yunqing Xia, and Ruifeng Xu. “Clique Percolation Method for Finding Naturally Cohesive and Overlapping Document Clusters”. In: *ICCPOL*. 2006, pages 97–108 (cited on page 25).

[36] Alain Gély, Lhouari Nourine, and Bachir Sadi. “Enumeration aspects of maximal cliques and bicliques”. In: *Discrete Applied Mathematics* 157.7 (2009), pages 1447–1459 (cited on page 25).

[37] Liqiang Geng and Howard J. Hamilton. “Interestingness measures for data mining: A survey”. In: *ACM Comput. Surv.* 38.3 (2006) (cited on page 67).

[38] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. “DB-CSC: A Density-Based Approach for Subspace Clustering in Graphs with Feature Vectors”. In: *ECML/PKDD (1)*. 2011, pages 565–580 (cited on pages 30, 37).

[39] Michel Habib. “Diameter and Center Computations in Networks”. In: *CTW*. 2009, pages 257–258 (cited on page 127).

[40] Phan Nhat Hai, Dino Ienco, Pascal Poncelet, and Maguelonne Teisseire. “Mining Time Relaxed Gradual Moving Object Clusters”. In: *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. SIGSPATIAL ’12. 2012, pages 478–481 (cited on page 43).

[41] Jiawei Han and Yongjian Fu. “Mining Multiple-Level Association Rules in Large Databases”. In: *IEEE Trans. Knowl. Data Eng.* 11.5 (1999), pages 798–804 (cited on page 77).

[42] Eyke Hüllermeier. “Association Rules for Expressing Gradual Dependencies”. In: *PKDD*. 2002, pages 200–211 (cited on page 43).

[43] Akihiro Inokuchi. “Mining Generalized Substructures from a Set of Labeled Graphs”. In: *ICDM*. 2004, pages 415–418 (cited on page 77).

[44] Akihiro Inokuchi and Takashi Washio. “A Fast Method to Mine Frequent Subsequences from Graph Sequence Data”. In: *ICDM*. 2008, pages 303–312 (cited on page 34).

[45] Akihiro Inokuchi and Takashi Washio. “GTRACE2: Improving Performance Using Labeled Union Graphs”. In: *PAKDD (2)*. 2010, pages 178–188 (cited on page 34).

[46] Akihiro Inokuchi and Takashi Washio. “Mining Frequent Graph Sequence Patterns Induced by Vertices”. In: *SDM*. 2010, pages 466–477 (cited on pages 23, 34).

[47] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. “An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data”. In: *PKDD*. 2000, pages 13–23 (cited on pages 23, 25).

[48] Paul Jaccard. “The Distribution of the Flora in the Alpine Zone”. In: *New Phytologist* 11.2 (1912), pages 37–50 (cited on page 47).

[49] Daxin Jiang and Jian Pei. “Mining frequent cross-graph quasi-cliques”. In: *ACM Trans. Knowl. Discov. Data (TKDD)* 2.4 (2009), pages 1–42 (cited on pages 16, 25, 49).

[50] Ruoming Jin, Scott McCallen, and Eivind Almaas. “Trend Motif: A Graph Mining Approach for Analysis of Dynamic Complex Networks”. In: *ICDM*. 2007, pages 541–546 (cited on pages 23, 37).

[51] Arijit Khan, Xifeng Yan, and Kun-Lung Wu. “Towards proximity pattern mining in large graphs”. In: *SIGMOD Conference*. 2010, pages 867–878 (cited on pages 27, 31).

[52] Michihiro Kuramochi and George Karypis. “Frequent Subgraph Discovery”. In: *ICDM*. 2001, pages 313–320 (cited on page 25).

[53] Michihiro Kuramochi and George Karypis. “GREW-A Scalable Frequent Subgraph Discovery Algorithm”. In: *ICDM*. 2004, pages 439–442 (cited on page 32).

[54] Michihiro Kuramochi and George Karypis. “Finding Frequent Patterns in a Large Sparse Graph”. In: *Data Min. Knowl. Discov. (DMKD)* 11.3 (2005), pages 243–271 (cited on page 16).

[55] Sergei O. Kuznetsov. “On stability of a formal concept”. In: *Ann. Math. Artif. Intell.* 49.1-4 (2007), pages 101–115 (cited on page 67).

[56] Mayank Lahiri and Tanya Y. Berger-Wolf. “Mining Periodic Behavior in Dynamic Social Networks”. In: *ICDM*. 2008, pages 373–382 (cited on page 32).

[57] Mayank Lahiri and Tanya Y. Berger-Wolf. “Periodic subgraph mining in dynamic networks”. In: *Knowl. Inf. Syst.* 24.3 (2010), pages 467–497 (cited on pages 23, 32).

[58] Matthieu Latapy and Clémence Magnien. “Complex Network Measurements: Estimating the Relevance of Observed Properties”. In: *INFOCOM*. 2008, pages 1660–1668 (cited on page 65).

[59] Matthijs van Leeuwen and Antti Ukkonen. “Discovering Skylines of Subgroup Sets”. In: *ECML/PKDD (3)*. 2013, pages 272–287 (cited on page 92).

[60] Guimei Liu and Limsoon Wong. “Effective Pruning Techniques for Mining Quasi-Cliques”. In: *ECML/PKDD (2)*. 2008, pages 33–49 (cited on pages 25, 49).

[61] Kazuhisa Makino and Takeaki Uno. “New Algorithms for Enumerating All Maximal Cliques”. In: *SWAT*. 2004, pages 260–272 (cited on page 25).

[62] Nikos Mamoulis, Huiping Cao, George Kollios, Marios Hadjieleftheriou, Yufei Tao, and David W. Cheung. “Mining, Indexing, and Querying Historical Spatiotemporal Data”. In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’04. New York, NY, USA: ACM, 2004, pages 236–245 (cited on page 15).

[63] Heikki Mannila and Hannu Toivonen. “Levelwise Search and Borders of Theories in Knowledge Discovery”. In: *Data Min. Knowl. Discov.* 1.3 (1997), pages 241–258 (cited on pages 15, 16).

[64] Shinichi Morishita and Jun Sese. “Traversing Itemset Lattice with Statistical Metric Pruning”. In: *PODS*. 2000, pages 226–236 (cited on page 67).

[65] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. “Mining Cohesive Patterns from Graphs with Feature Vectors”. In: *SDM*. 2009, pages 593–604 (cited on pages 16, 23, 27).

[66] Pierre-Nicolas Mougel, Christophe Rigotti, and Olivier Gandrillon. “Finding Collections of k-Clique Percolated Components in Attributed Graphs”. In: *PAKDD (2)*. 2012, pages 181–192 (cited on pages 16, 23, 25, 29, 30).

[67] Pierre-Nicolas Mougel, Christophe Rigotti, Marc Plantevit, and Olivier Gandrillon. “Finding maximal homogeneous clique sets”. In: *KAIS*. 2013, pages 1–30 (cited on page 30).

[68] M. Nanni, B. Kuijpers, C. Körner, M. May, and D. Pedreschi. “Spatiotemporal Data Mining”. In: *Mobility, Data Mining and Privacy*. Edited by Fosca Giannotti and Dino Pedreschi. Springer Berlin Heidelberg, 2008, pages 267–296 (cited on page 15).

[69] Benjamin Négrevergne, Anton Dries, Tias Guns, and Siegfried Nijssen. “Dominance Programming for Itemset Mining”. In: *ICDM*. 2013, pages 557–566 (cited on page 92).

[70] Benjamin Négrevergne, Alexandre Termier, Marie-Christine Rousset, and Jean-François Méhaut. “Para Miner: a generic pattern mining algorithm for multi-core architectures”. In: *Data Min. Knowl. Discov.* 28.3 (2014), pages 593–633 (cited on page 127).

[71] Kim-Ngan Nguyen, Loïc Cerf, Marc Plantevit, and Jean-François Boulicaut. “Discovering Inter-Dimensional Rules in Dynamic Graphs”. In: *NyNaK*. 2010 (cited on page 35).

[72] Kim-Ngan Nguyen, Loïc Cerf, Marc Plantevit, and Jean-François Boulicaut. “Multidimensional Association Rules in Boolean Tensors”. In: *SDM*. 2011, pages 570–581 (cited on page 35).

[73] Kim-Ngan Nguyen, Loïc Cerf, Marc Plantevit, and Jean-François Boulicaut. “Discovering descriptive rules in relational dynamic graphs”. In: *Intell. Data Anal.* 17.1 (2013), pages 49–69 (cited on pages 23, 35).

[74] Siegfried Nijssen and Joost N. Kok. “Frequent graph mining and its application to molecular databases”. In: *Systems, Man and Cybernetics (SMC)*. Volume 5. 2004, pages 4571–4577 (cited on page 16).

[75] Siegfried Nijssen and Joost N. Kok. “The Gaston Tool for Frequent Subgraph Mining”. In: *Electr. Notes Theor. Comput. Sci.* 127.1 (2005), pages 77–87 (cited on page 25).

[76] Günce Keziban Orman, Vincent Labatut, Marc Plantevit, and Jean-François Boulicaut. “Une méthode pour caractériser les communautés des réseaux dynamiques à attributs”. In: *EGC*. 2014, pages 101–112 (cited on page 37).

[77] Nikhil R Pal and Sankar K Pal. “A review on image segmentation techniques”. In: *Pattern Recognition* 26.9 (1993), pages 1277–1294 (cited on page 106).

[78] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. “Uncovering the overlapping community structure of complex networks in nature and society”. In: *Nature* 435.7043 (2005), pages 814–818 (cited on page 25).

[79] Raj Kumar Pan, Mikko Kivelä, Jari Saramäki, Kimmo Kaski, and János Kertész. "Explosive percolation on real-world networks". In: *CoRR* abs/1010.3171 (2010) (cited on page 25).

[80] Apostolos N. Papadopoulos, Apostolos Lyritsis, and Yannis Manolopoulos. "SkyGraph: an algorithm for important subgraph discovery in relational graphs". In: *Data Min. Knowl. Discov.* 17.1 (2008), pages 57–76 (cited on pages 16, 18, 92).

[81] Claude Pasquier, Jérémy Sanhes, Frédéric Flouvat, and Nazha Selmaoui-Folcher. "Frequent Pattern Mining in Attributed Trees". In: *PAKDD (1)*. 2013, pages 26–37 (cited on page 28).

[82] Ruggero G. Pensa, Francesca Cordero, Céline Rouveirol, and Rushed Kanawati. "Guest Editorial". In: *Intell. Data Anal.* 17.1 (2013), pages 1–3 (cited on page 15).

[83] Adriana Prado, Baptiste Jeudy, Élisa Fromont, and Fabien Diot. "Mining spatiotemporal patterns in dynamic plane graphs". In: *Intell. Data Anal.* 17.1 (2013), pages 71–92 (cited on page 36).

[84] Adriana Prado, Marc Plantevit, Céline Robardet, and Jean-François Boulicaut. "Mining Graph Topological Patterns: Finding Covariations among Vertex Descriptors". In: *IEEE Trans. Knowl. Data Eng.* 25.9 (2013), pages 2090–2104 (cited on pages 16, 31, 43).

[85] Ronald C. Read and Derek G. Corneil. "The graph isomorphism disease". In: *Journal of Graph Theory* 1.4 (1977), pages 339–363 (cited on page 15).

[86] Céline Robardet. "Constraint-Based Pattern Mining in Dynamic Graphs". In: *ICDM*. 2009, pages 950–955 (cited on pages 16, 17, 23, 33).

[87] John F. Roddick and Myra Spiliopoulou. "A Bibliography of Temporal, Spatial and Spatio-temporal Data Mining Research". In: *SIGKDD Explor. Newsl.* 1.1 (June 1999), pages 34–38 (cited on page 15).

[88] Willy Ugarte Rojas, Patrice Boizumault, Samir Loudni, Bruno Crémilleux, and Alban Lepailleur. "Mining (Soft-) Skypatterns Using Dynamic CSP". In: *CPAIOR*. 2014, pages 71–87 (cited on pages 92, 128).

[89] Jun Sese, Mio Seki, and Mutsumi Fukuzaki. "Mining networks with shared items". In: *CIKM*. 2010, pages 1681–1684 (cited on pages 28, 30).

[90] Haichuan Shang, Ying Zhang, Xuemin Lin, and Jeffrey Xu Yu. "Taming verification hardness: an efficient algorithm for testing subgraph isomorphism". In: *PVLDB* 1.1 (2008), pages 364–375 (cited on page 25).

[91] Prakash Shelokar, Arnaud Quirin, and Oscar Cordón. "MOSubdue: a Pareto dominance-based multiobjective Subdue algorithm for frequent subgraph mining". In: *Knowl. Inf. Syst.* 34.1 (2013), pages 75–108 (cited on pages 18, 92).

[92] Arlei Silva, Wagner Meira Jr., and Mohammed J. Zaki. "Mining Attribute-structure Correlated Patterns in Large Attributed Graphs". In: *PVLDB* 5.5 (2012), pages 466–477 (cited on pages 16, 23, 25, 29).

[93] Arlei Silva, Wagner Meira Jr., and Mohammed J. Zaki. "Structural Correlation Pattern Mining for Large Graphs". In: *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*. MLG '10. Washington, D.C.: ACM, 2010, pages 119–126 (cited on page 29).

[94] Kelvin Sim, Jinyan Li, Vivekanand Gopalkrishnan, and Guimei Liu. “Mining maximal quasi-bicliques: Novel algorithm and applications in the stock market and protein networks”. In: *Statistical Analysis and Data Mining* 2.4 (2009), pages 255–273 (cited on page 25).

[95] Arnaud Soulet and Bruno Crémilleux. “An Efficient Framework for Mining Flexible Constraints”. In: *PAKDD*. 2005, pages 661–671 (cited on page 16).

[96] Arnaud Soulet, Chedy Raïssi, Marc Plantevit, and Bruno Crémilleux. “Mining Dominant Patterns in the Sky”. In: *ICDM*. 2011, pages 655–664 (cited on pages 18, 66, 92).

[97] Ramakrishnan Srikant and Rakesh Agrawal. “Mining Sequential Patterns: Generalizations and Performance Improvements”. In: *EDBT*. 1996, pages 3–17 (cited on pages 17, 77).

[98] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005 (cited on page 47).

[99] Kilian Thiel and Michael R. Berthold. “Node Similarities from Spreading Activation”. In: *Bisociative Knowledge Discovery*. 2012, pages 246–262 (cited on pages 48, 58).

[100] Etsuji Tomita, Yoichi Sutani, Takanori Higashi, and Mitsuo Wakatsuki. “A Simple and Faster Branch-and-Bound Algorithm for Finding a Maximum Clique with Computational Experiments”. In: *IEICE Transactions* 96-D.6 (2013), pages 1286–1298 (cited on page 25).

[101] Hanghang Tong, Christos Faloutsos, Brian Gallagher, and Tina Eliassi-Rad. “Fast best-effort pattern matching in large attributed graphs”. In: *KDD*. 2007, pages 737–746 (cited on page 26).

[102] Charalampos E. Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A. Tsiarli. “Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees”. In: *KDD*. 2013, pages 104–112 (cited on page 16).

[103] Bianca Wackersreuther, Peter Wackersreuther, Annahita Oswald, Christian Böhm, and Karsten M. Borgwardt. “Frequent Subgraph Discovery in Dynamic Networks”. In: *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*. MLG ’10. ACM, 2010, pages 155–162 (cited on page 32).

[104] Jianyong Wang, Zhiping Zeng, and Lizhu Zhou. “CLAN: An Algorithm for Mining Closed Cliques from Large Dense Graph Databases”. In: *Int. Conf. on Data Engineering (ICDE)*. 2006, page 73 (cited on page 16).

[105] Ingo Wegener. *Complexity theory - exploring the limits of efficient algorithms*. Springer, 2005, pages I–XI, 1–308 (cited on page 25).

[106] Xifeng Yan and Jiawei Han. “gSpan: Graph-Based Substructure Pattern Mining”. In: *ICDM*. 2002, pages 721–724 (cited on pages 16, 23, 25).

[107] Chang Hun You, Lawrence B. Holder, and Diane J. Cook. “Learning patterns in the dynamics of biological networks”. In: *KDD*. 2009, pages 977–986 (cited on pages 17, 33).

[108] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. “Out-of-core coherent closed quasi-clique mining from large dense graph databases”. In: *ACM Trans. Database Syst.* 32.2 (2007), page 13 (cited on pages 25, 49).

[109] Shijie Zhang, Meng Hu, and Jiong Yang. “TreePi: A Novel Graph Indexing Method”. In: *ICDE*. 2007, pages 966–975 (cited on page 25).

[110] Peixiang Zhao, Jeffrey Xu Yu, and Philip S. Yu. “Graph Indexing: Tree + Delta >= Graph”. In: *VLDB*. 2007, pages 938–949 (cited on page 25).