Mining useful patterns in attributed graphs

Anes Bendimerad

N° d'ordre NNT : 2019LYSEI058

Thèse de doctorat de l'Université de Lyon
opérée au sein de
L'INSA de Lyon

Ecole Doctorale N° 512
Mathématiques et Informatique (InfoMaths)

Spécialité / discipline de doctorat : Informatique

Soutenue publiquement le 05/09/2019 par
Ahmed Anes Bendimerad

Mining Useful Patterns in Attributed Graphs

Devant le jury composé de:

Pr. Aristides Gionis	Aalto University	Rapporteur
Pr. Marie-Christine Rousset	Université Grenoble Alpes	Rapporteure
Pr. Tijl De Bie	Ghent University	Examinateur
Pr. Alexandre Termier	Université de Rennes 1	Examinateur
Dr. Siegfried Nijssen	Université Catholique de Louvain	Examinateur
Pr. Céline Robardet	INSA-Lyon	Directrice de thèse
Dr. Marc Plantevit	Université Claude Bernard Lyon 1	Co-directeur de thèse

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** http://www.edchimie-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX directeur@edchimie-lyon.fr
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** http://edeea.ec-lyon.fr Sec. : M.C. HAVGOUDOUKIAN ecole-doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 gerard.scorletti@ec-lyon.fr
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** http://e2m2.universite-lyon.fr Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES secretariat.e2m2@univ-lyon1.fr	M. Philippe NORMAND UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX philippe.normand@univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** http://www.ediss-lyon.fr Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE secretariat.ediss@univ-lyon1.fr	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 emmanuelle.canet@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** http://edinfomaths.universite-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 zamboni@maths.univ-lyon1.fr
Matériaux	**MATÉRIAUX DE LYON** http://ed34.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 jean-yves.buffiere@insa-lyon.fr
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** http://edmega.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction mega@insa-lyon.fr	M. Jocelyn BONJOUR INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX jocelyn.bonjour@insa-lyon.fr
ScSo	**ScSo*** http://ed483.univ-lyon2.fr Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 veronique.cervantes@univ-lyon2.fr	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 christian.montes@univ-lyon2.fr

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Abstract

In this thesis, we address the problem of pattern discovery in vertex-attributed graphs. This kind of structure consists of a graph augmented with attributes associated to vertices. Vertex-attributed graphs provide a powerful abstraction that can be used to represent many datasets in an intuitive manner. Mining these graphs can be very useful for many applications, such as analyzing social networks, biological networks, the World Wide Web, etc. Several methods have been proposed to identify patterns in these structures. Generally, these methods define a pattern as a subgraph whose vertices satisfy some structural constraints (e.g., density, connectivity) and have a subset of attributes with homogeneous values.

When mining vertex-attributed graphs, the principled integration of both graph and attribute data poses two important challenges. First, we need to define a pattern syntax (the abstract form of patterns) that is intuitive and lends itself to efficient search. A pattern being intuitive means that it can be easily interpreted and assimilated by the user. Considering that a pattern is generally defined over a subgraph, a pattern can be often huge in terms of vertices, which makes it difficult to grasp. Thus, the assimilation cost of a pattern is an important question that needs to be addressed. The second challenge is the formalization of the pattern interestingness. A pattern is generally relevant if it depicts some local properties that are somehow exceptional, otherwise, it will be already expected from the overall properties of the graph. Furthermore, the interestingness of patterns is subjective in practice, i.e., it significantly depends on the final user, her background knowledge and her preferences. A user would consider that a pattern is useful if it brings some new knowledge to her, especially if this pattern informs about some features or topics that usually interest this user. Another common problem related to the interestingness of patterns is the redundancy issue in the result set. In other terms, a data mining approach may return a set of patterns that give redundant information, because these patterns cover very overlapping parts of vertices and attributes. Information redundancy can be also due to some semantic relation between different attributes, such as attribute hierarchies. For example, knowing that a community of a social network is characterized by a high interest in “rock music” makes it less informative that it also has a high interest in “music”, because “rock music” is a subtype “music”. Consequently, the quality of patterns depends on many different factors.

We address these challenges for the problem of mining attributed graphs. More precisely, we first introduce the task of discovering exceptional attributed subgraphs, which is rooted in the Subgroup Discovery framework. The goal is to identify connected subgraphs whose vertices share characteristics that distinguish them from the rest of the graph. Then, we propose methods that aim to take into account the user and the domain knowledge when assessing the interestingness of patterns. We design a

method that makes it possible to incorporate user's background knowledge and pattern's assimilation cost. This method is able to identify patterns that are both unexpected (thus informative) and easy to interpret. To ease the assimilation, alternative descriptions of exceptional attributed subgraphs are provided. Furthermore, we propose another graph mining approach that integrates user's preferences. This method exploits an interactive process with the user to bias the pattern interestingness. It has been defined for the task of geo-located event detection in social media. Then, we design an approach that is able to incorporate hierarchical attribute dependencies into the pattern interestingness, which allows to avoid redundancy related to this kind of semantic relations between attributes. In other terms, when the attributes are organized as a hierarchy, this method is able to account for the inference that the user would make about some attribute values when she is informed about values of other attributes. Finally, we conclude this thesis by discussing some research perspectives.

Résumé

Nous adressons le problème de découverte de motifs dans les graphes attribués. Cette structure de données correspond à un graphe qui est augmenté par des attributs associés aux sommets. Elle permet de modéliser efficacement et intuitivement une large variété de bases de données réelles. L'analyse de ce type de graphes peut offrir une grande opportunité pour extraire des informations utiles et actionnables, par exemple, l'analyse des réseaux sociaux, réseaux biologiques, réseaux internet, etc.

La fouille de graphes attribués nécessite des méthodes qui prennent en compte au même temps la structure du graphe et les attributs décrivant les sommets, et cela génère deux défis. Premièrement, il est important de définir un langage de motifs intuitif sur lequel on peut appliquer des stratégies de recherche efficaces. Un motif étant intuitif signifie qu'il peut être facilement interprété et compris par l'utilisateur. Sachant qu'un motif est généralement défini sur un sous-graphe, il peut donc être immense en nombre de sommets, ce qui le rend difficile à comprendre. Le coût d'assimilation du motif est donc une question importante qui doit être adressée. Le deuxième défi est la formalisation de la mesure de qualité (pertinence) des motifs. Un motif local est généralement pertinent s'il décrit des propriétés locales distinctives, autrement, ce motif serait déjà attendu en regardant les propriétés globales du graphe. Par ailleurs, la qualité d'un motif est subjective, i.e., elle dépend significativement de l'utilisateur final, de ses connaissances antérieurs sur les données et de ses préférences. Généralement, un utilisateur considère qu'un motif est utile s'il lui fournit de nouvelles connaissances, particulièrement si ce motif lui informe sur des caractéristiques ou des sujets qui intéressent habituellement l'utilisateur. Un autre problème lié à la qualité des motifs est la redondance. En d'autres termes, une méthode de fouille de données peut retourner un ensemble de motifs qui donnent des informations redondantes, par exemple, des motifs peuvent couvrir des parties significativement superposées de sommets et d'attributs. La redondance d'information peut être aussi due aux relations sémantiques entre les attributs, comme les hiérarchies d'attributs. Par exemple, dans un réseau social, si on sait déjà qu'une communauté est caractérisée par un grand intérêt lié à la "musique du rock", caractériser cette communauté encore par "musique" serait redondant, car "musique du rock" est un sous-type de "musique". Dans cette thèse, nous adressons ces différents défis pour le problème de la fouille de graphes attribués. Plus précisément, nous définissons de nouveaux langages de motifs, des mesures de qualités, des algorithms pour la fouille de graphes attribués. On réalise aussi des études empiriques approfondies pour évaluer la pertinence de ces contributions.

Publications presented in the thesis

The contributions presented in this thesis appear in the following publications:

International journals

- Anes Bendimerad, Jefrey Lijffijt, Marc Plantevit, Céline Robardet, and Tijl De Bie. SIAS-Miner: Mining subjectively interesting attributed subgraphs. *Data Mining and Knowledge Discovery.*

- Anes Bendimerad, Marc Plantevit, Céline Robardet, and Sihem Amer-Yahia. User-driven geolocated event detection in social media. *IEEE Transactions on Knowledge and Data Engineering, 2019.*

- Anes Bendimerad, Marc Plantevit, and Céline Robardet. Mining exceptional closed patterns in attributed graphs. *Knowledge and Information Systems, 56(1):1–25, 2018.*

International conferences

- Anes Bendimerad, Jefrey Lijffijt, Marc Plantevit, Céline Robardet and Tijl De Bie. Contrastive antichains in hierarchies. *In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.*

- Anes Bendimerad, Marc Plantevit, and Céline Robardet. Unsupervised exceptional attributed sub-graph mining in urban data. *In IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain, pages 21–30, 2016.*

International workshops

- Anes Bendimerad, Ahmad Mel, Jefrey Lijffijt, Marc Plantevit, Céline Robardet, and Tijl De Bie. Mining subjectively interesting attributed subgraphs. *In 15th International Workshop on Mining and Learning with Graphs, held with SIGKDD, 2018.*

Other publications

- Anes Bendimerad, Rémy Cazabet, Marc Plantevit, and Céline Robardet. Contextual subgraph discovery with mobility models. *In Complex Networks & Their Applications VI., pages 477–489, 2017.*
- Mehdi Kaytoue, Marc Plantevit, Albrecht Zimmermann, Anes Bendimerad, and Céline Robardet. Exceptional contextual subgraph mining. *Machine Learning, 106(8):1171–1211, 2017.*

PROTOTYPES

CENERGETICS and EXCESS. These algorithms are designed for the task of mining exceptional subgraphs in vertex-attributed graphs. They aim to identify connected subgraphs whose vertices have some attribute characteristics that distinguish them from the rest of the graph. They are presented in details in Chapter 3. CENERGETICS is an exhaustive algorithm that identifies the complete set of closed exceptional subgraphs, while EXCESS is a heuristic algorithm that uses an output space sampling strategy to enable time budget analysis.

- `https://github.com/AnesBendimerad/ClosedExceptionalSubgraphMining`

SIAS-Miner. This algorithm mines CSEA patterns, i.e., Cohesive Subgraphs with Exceptional Attributes, which is an improved pattern syntax of exceptional attributed subgraph patterns introduced in Chapter 3. The interestingness of these patterns is assessed using a model built upon the subjective interestingness framework of De Bie [59]. Chapter 4 presents this approach.

- `https://www.dropbox.com/sh/906wnv5a8c9ecx1/AACpUf1_0sa6pC0vtwlg_Q33a?dl=0`

SIGLER-Cov and SIGLER-Samp. These algorithms address the problem of user-driven event detection in social media. They perform this task based on an exceptional attributed subgraph mining approach. They enable the incorporation of user's preferences in the interestingness model employed to evaluate the quality of detected events. SIGLER-Cov is a branch and bound algorithm that returns a result set with a coverage guarantee, while SIGLER-Samp is an algorithm based on output space sampling strategy. These methods are presented in Chapter 5.

- `https://github.com/AnesBendimerad/User-driven-geolocated-event-detection`

MICA-Miner. This algorithm is designed for the discovery of contrastive antichains in hierarchical attributes, i.e., a particular subset of attributes that characterize a dataset with their distinctive values. The measure used to evaluate the quality of contrastive antichains is also rooted on the Subjective Interestingness framework of De Bie [59]. Chapter 6 details this method.

- `https://bitbucket.org/ghentdatascience/mica-miner/`

Acknowledgements

First of all, I warmly thank Aristides Gionis, Professor at Aalto University, and Marie-Christine Rousset, Professor at the University of Grenoble Alpes, for accepting to review my PhD thesis. I thank them for all the time and efforts they've engaged to read the manuscript and write the reviews. I deeply thank as well Professor Tijl De Bie, Professor Alexandre Termier, Doctor Sigfried Nijssen, and Bertrand Duqueroie to be part of my PhD committee.

I'd like to express my sincere gratitude to my advisors Marc Plantevit and Céline Robardet, for the continuous support of my PhD study and related research, for their patience, motivation, and trust. I've learned a lot from them about data mining and research, but also about work and my own abilities in this field. I also thank my scientific tutor Atilla Baskurt for ensuring that the collaboration with Thales goes perfectly well, and for providing his support every needed time. Many thanks also to DM2L members, among them Jean-François Boulicaut, Mehdi Kaytoue, and Remy cazabet, for all their precious advice, support, and pleasant discussions.

I'd like to thank people I collaborated with, during my PhD. I express my deepest appreciation to Tijl De Bie and Jefrey Lijffijt and their group from Ghent University, for their very warm welcome, enthusiasm, for all the pleasant time I spent with them, and for the significant amount of things that I learned from them. I also thank Sihem Amer-yahia for her great support, I've really appreciated collaborating with her.

I warmly thank my colleagues and my friends: Tarek, Hind, Chabha, Maelle, Marie, Amine, Lucas, Julien, Guillaume, Romain, Diana, Florian, etc., for all the fun and support we had, and for the interesting (or not, but at least pleasant) discussions during lunch time. A special thank goes to special friends: Aimene, Adnene, Mohamed who I know from the hard ESI times.

Last but not the least, I owe a huge dept of gratitude to my family (my parents, brothers and sisters) and Wissam for supporting me throughout all the ups and downs during my PhD, and my life in general.

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Context

Data mining is an interdisciplinary subfield of computer science and statistics, whose goal is to extract relevant information from datasets. It is concerned with discovering patterns that elicit some new knowledge which is nontrivial, implicit, potentially useful and actionable. For instance, such a knowledge can be used afterhand to enhance the decision support in different application domains. Data mining has been highly motivated by the current exponential growth of data in the world, such as the one generated by social networks, genomics and proteomics datasets, GPS track records, etc. Analyzing these data can significantly help to solve many problems. For example, mining social networks makes it possible to identify controversial topics and understand how they evolve. In biology and bioinformatics, data mining can be exploited to explain the role of different genes and relations between them. The research results presented in this manuscript are mostly related to the extraction of patterns that describe some areas of a city and the events that take place there. This was motivated by the collaboration with the Thales DSC Theresis Innovation Laboratory.

1.1.1 Pattern mining framework

Research in pattern mining can be easily summarized from an inductive database perspective [148] as the computation of the theory Th defined as:

$$Th(D, \mathcal{L}, \mathcal{C}) = \{P \in \mathcal{L} \mid \mathcal{C}(P, D) \text{ is true}\}. \tag{1.1}$$

It consists in the enumeration of all patterns from a language $\mathcal{L}$ that fulfill a user-defined constraint $\mathcal{C}$ in a given database D. Let us illustrate this framework on the popular problem of frequent itemset mining introduced by Agrawal et al. [7]. A dataset $D = (I, O)$ is defined by a set of objects $O = \{o_1, ..., o_m\}$,

where each object corresponds to a subset of items from $I = \{i_1, ..., i_n\}$, i.e., for all $o \in O$ we have $o \subseteq I$. An example of such a dataset is given in Tab 1.1. It represents sale transactions in a marketplace: transactions are modeled as objects in O and market products are represented by items in I. The frequent itemset mining problem is concerned with finding itemsets $P \in 2^I$ that frequently appear in D. The frequency $freq(P, D)$ of a pattern P is the percentage of objects that contain the itemset P, that is $freq(P, D) = \frac{|\{o \in O | P \subseteq o\}|}{|O|}$. In the context of analyzing sale transactions, this task identifies set of products that are bought frequently together and has application in crossmarketing.

O	i_1	i_2	i_3	i_4	i_5
o_1	1	0	1	0	0
o_2	1	1	1	0	0
o_3	0	0	1	1	0
o_4	1	0	1	0	0
o_5	1	0	1	1	1
o_6	0	0	0	1	0

Table 1.1: Example of dataset of objects $O = \{o_1, ..., o_6\}$ described by items $I = \{i_1, ..., i_5\}$. A value 1 (resp. 0) at the intersection of a row and a column means that the object corresponding to this row contains (resp. does not contain) the item corresponding to this column.

This founded framework has been instantiated in numerous data mining tasks by specifying a particular pattern language, an interestingness measure that expresses the user's need, and an algorithm that makes possible the computation of such theories in real-world datasets. Let us look in more details at the important aspects of this framework. Interested readers can also refer to Soulet [195] who presents an overview of the topics and problems investigated by the data mining community during the last two decades.

Language. The pattern language is the domain of definition of patterns that specifies what the user is looking for. For example, in the frequent itemset mining problem, the pattern language is $\mathcal{L} = 2^I$, the language of itemsets. Many different pattern languages have been defined for more complex structures of data, and a myriad of methods have been proposed to extract patterns from these structures, such as numerical data (e.g., interval patterns [120], gradual patterns [65]), graphs (e.g., frequent subgraphs [48, 126, 163, 222], cliques [11, 49, 89, 146], quasi-cliques [143, 169, 226]), sequential data (e.g., sequence of itemsets [6, 198], strings [83, 132, 177], episodes [147, 150]), trees (e.g., subtrees [55, 56, 162, 201–203, 225]) , spatio-temporal data (e.g., trajectories [90], spatio-temporal sequential patterns [51]). Defining expressive pattern languages remains a challenging problem, especially for complex data types such as graphs and spatio-temporal data.

Interestingness. Defining what makes a pattern useful or interesting for a user is also a challenging task. Originally, as expressed in equation 1.1, the interestingness of a pattern was assumed to be expressed by a set of constraints involving some measures (i.e., threshold based constraints). For example, the

frequent itemset mining problem uses the pattern frequency $freq(P)$ as interestingness measure: if $freq(P)$ fulfills a minimum threshold, P is then reported to the user. While this measure is simple and easy to interpret, it has several limitations. For example, if two items i_1 and i_3 are frequent, it is statistically expected that the pattern $\{i_1, i_3\}$ is also frequent. Informing the user that $\{i_1, i_3\}$ is frequent does not bring a new knowledge for her in this case. Hence, only considering the frequency does not make it possible to provide insightful results. Based on this observation, a large variety of constraints and quality measures have been designed to assess the interestingness of patterns. A first line of research consisted of evaluating the quality of a pattern with respect to a class variable. The patterns that well describe a class of such a variable are preferred by defining specific constraints (e.g., threshold, top k) on measures evaluating how discriminant is a pattern such as the growth rate [66], or the Weighted Relative Accuracy measure (WRAcc) [130]. In another line of research, several interestingness measures have been proposed to assess the statistical significance of patterns, using some specific techniques such as swap randomization [92] and permutation testing [145]. Finally, more recent work aims to integrate user's preferences into the mining process [30, 72, 220], or to take into account the user background knowledge in the pattern selection process. These measures favor patterns that are surprising when contrasted to the user prior knowledge [59, 60]. In doing so, these recent measures require extending the framework formalized by equation 1.1, in order to integrate the user and make the process interactive.

In general terms, the pattern interestingness depends on several factors such as the goal of the mining task, the application domain, the user prior knowledge, etc. This makes the definition of pattern interestingness a challenging topic. For this reason, huge effort has been given by the data mining community to address this question.

Mining algorithm. Once the language and the pattern interestingness defined, it remains to design an algorithm that computes the theory $Th(D, \mathcal{L}, \mathcal{C})$. Designing such an *exhaustive method* requires to study the pattern language properties and define specialization relation which makes it possible to perform a complete and hopefully non redundant enumeration of the search space. These methods usually exploit optimization techniques to prune unpromising parts of the search space. User-defined constraints play an important role when designing pruning techniques, as they can be exploited to achieve computational feasability. Many types of constraints have been studied, e.g., (anti-)monotone [7], convertible constraints [168], loose anti-monotone [39, 206], optimistic estimates [216]. Each of them is associated with specific optimization techniques. Several efficient parallel approaches have been also proposed to improve the time performance of the pattern discovery task by exploiting multicore architectures [65, 123, 160, 200]. Moreover, instead of returning all the patterns satisfying the user-specified constraints, some exhaustive methods propose to only extract the top-k patterns with respect to the interestingness measure, with $k \in \mathbb{N}$ a user-specified parameter. Furthermore, to reduce the size of the search space and the redundancy among the result set, algorithms have been designed to return only condensed representations of patterns, such as closed patterns [207, 223], maximal patterns [94, 112], generators (free-sets) [45]. While exhaustive approaches have the guarantee of completeness, their scalability is limited as the size of the search space is generally exponential with respect to the size of

the dataset.

Thus, heuristic algorithms have been proposed to deal with the most difficult mining tasks. They focus on the approximate search of interesting patterns. Several optimization strategies have been used for exploring the pattern language, such as hill climbing [208], beam search [87, 129], evolutionary algorithms [62, 165, 179], output space sampling [37, 38, 96, 153, 154]. These algorithms scale better than exhaustive methods but they do not find the exact result set, and most of them do not provide guarantees about the result quality.

Finally, *anytime algorithms* have been recently proposed for mining patterns [22, 43, 232]. These algorithms yield progressively patterns whose quality improves over times, they can be interrupted anytime, and they often have the guarantee to return the exact result set if enough time is provided. Particularly for interval patterns, Belfodil et al. [22] provide a guarantee bounding the error on the top pattern quality when the algorithm is interrupted before the end of its execution.

1.1.2 Describing cities and their events

This thesis is part of a collaboration between the AS& BSim team (Adaptive System & Biomimetic Simulation) of Thales DSC Theresis Innovation Laboratory and the LIRIS. AS& BSim is developing an infrastructure simulation application, named SE-Star (Synthetic Environment Components Suite), which aims to realistically reproduce the behavior of users of some infrastructures through a biomimetic multi-agents model. Setting some new simulation is a heavy task. The goal of this collaboration is to define methods to help setting and monitoring simulations. One of the case studies we looked at is a simulation that works well in one city and we want to set up a similar simulation in another city. It is then necessary to automatically characterize the areas of the new city, as well as the dynamics of the events that occur there, and to find the similarities with the city that we are able to well simulate. From this information, a transfer of knowledge can be done in the setting of the simulations. Such information can be obtained by mining several heterogeneous sources of data (e.g., social networks, demographic data, open data). It is then necessary to be able to mine heterogeneous data using a pattern domain that handles this heterogeneity. This led us to develop several pattern domains on attributed graphs.

1.2 Mining a rich data type

As previously stated, many methods have been designed to mine patterns in complex data structures such as graphs and sequences. Among them, vertex-attributed graphs have attracted a lot of attention in the data mining community. In fact, they offer a powerful mathematical abstraction that is able to represent many datasets in an intuitive manner, particularly when we have objects that are simultanuously interacting and described by some attributes. Formally, a vertex-attributed graph $G = (V, E, A)$ is defined by a set of vertices V, a set of edges between vertices $E \subseteq V \times V$, and a set of attributes A that describe vertices. Fig 1.1 shows an example where this structure is used to represent a set of social network users with their relationships and their membership in groups by topic. For example, the vertex

v_1 represents a user who is a member of 2 musical groups, 5 political groups, and he interacts with users v_2, v_3 and v_4. Edges can be used to represent interactions such as friendship relations. In this example, attributes are numerical and they represent counts. In general, several types of attributes can be used, for example Boolean, categorical, or numerical.

Mining this kind of graphs can be very useful for many applications. In a graph representing a social network in the same way as in Fig 1.1, mining this structure allows one to identify and describe communities of individuals sharing the same topics of interests. When a company targets a specific set of clients, understanding characteristics of their communities would allow to propose better marketing plans for their products and services. In the context of mining urban data, a city can be modelled as a graph whose vertices represent city areas described by attributes that indicate the prevalence of different kinds of facilities (outdoor facilities such as parks, food places such as restaurants, colleges, etc.), and edges represent the geographic closeness of areas. Mining this graph makes it possible to identify city blocks that are geographically close and consistent in terms of service offerings. These findings can then be used to recommend areas to people that move into a new city [77] while wishing to keep the characteristics of their previous neighborhood.

Several methods have been designed to extract patterns from vertex-attributed graphs [86, 98, 122, 156, 190]. For example, the pioneering work of Moser et al. [156] presents a method to mine dense homogeneous subgraphs, i.e., subgraphs whose vertices are highly connected and share a large set of attributes. From a general perspective, a pattern usually corresponds to a subset of vertices $U \subseteq V$ satisfying some structural constraints (e.g., connectivity [86], density [98]), and having a subset of attributes with homogeneous values. While many methods have been proposed to address the problem of pattern discovery in this kind of structures, there is still a lot of challenges to solve. These challenges are related to the three aforementioned key components of pattern discovery (language, interestingness, and mining algorithm). In this thesis, we are interested in the problem of mining useful patterns in vertex-attributed graphs. Particularly, we study graphs where attributes are numerical and represent counts, as in Fig 1.1. We aim to address some specific challenges that are mainly - but not only - related to the definition of the pattern interestingness.

To provide an efficient and useful data mining approach, it is important to design a pattern language that is intuitive (easy to interpret), flexible, able to capture the properties of interest, and lends itself to efficient search. Satisfying simultanuously all these criteria can be challenging especially for complex structures such as attributed graphs. For example, methods designed on this kind of structure generally output patterns that are defined as set of vertices $U \subseteq V$ satisfying specific constraints. However, the size of U is sometimes huge, and the final user will not be able to read each of vertices in U when this pattern is presented to her. How to find a language that makes it easy for the user to assimilate patterns? Another bottleneck when mining vertex-attributed graphs is to come up with meaningful and interesting constraints on vertex-attributes. In other terms, what are the attribute-characteristics that the user would like to know about?

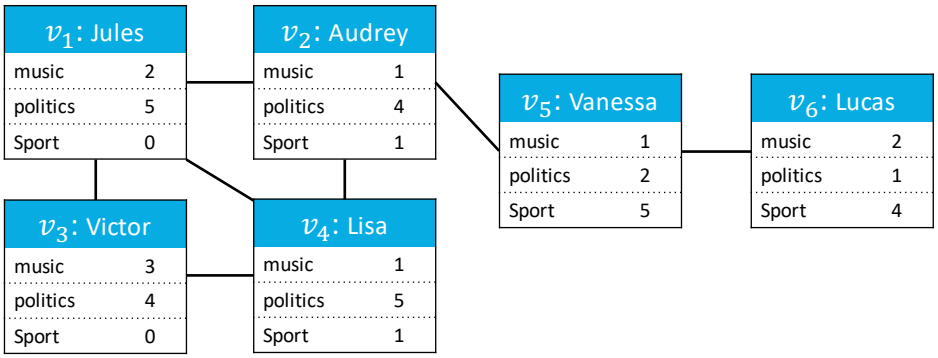


Figure 1.1: Toy vertex-attributed graph $G = (V, E, A)$ with the set of vertices $V = \{v_1, v_2, ..., v_6\}$, and numerical attributes $A = \{\text{music}, \text{politics}, \text{sport}\}$. Attributes depict the number of membership groups per topic for each vertex (individual).

1.3 Integrating user priors and preferences

Pattern discovery usually requires some interestingness to evaluate the relevance of patterns. A pattern is likely to be relevant when it depicts some local properties that significantly deviate from the structure of the whole dataset, otherwise this pattern would be already expected. In Fig 1.1, it is relevant to describe the community $\{v_1, v_2, v_3, v_4\}$ by a high prevalence of "politics" groups and a low presence of "sport" groups, because these features are characteristic of this specific community. However, most of vertex-attributed graph mining approaches focus on identifying subgraphs having homogeneous attribute values, i.e., they seek for regularities while underestimating discriminativity. The task of discovering patterns that discriminate a subset of data from the overall dataset has been efficiently formalized with Subgroup Discovery (SD) [125, 130] and Exceptional Model Mining (EMM) [71]. Many discriminatory measures have been proposed for SD and EMM to identify exceptional patterns. One of the most commonly used measures is the Weighted Relative Accuracy (WRAcc) [130]. Extending these methods and measures to identify exceptional patterns in vertex-attributed graphs is challenging, as it requires a principled integration of both graph structure and attribute data.

Furthermore, the interestingness of a pattern is subjective in practice, i.e., it significantly depends on the user. For the same data mining task, two users may have different judgments about the discovered patterns. This is highly related to their background knowledge, their preferences, and their target objective. For example, a pattern P can be deemed interesting by a user u_1 but not by u_2, because u_2 already has some prior knowledge that makes her expect this pattern, or because this pattern does not depict features that interest u_2. Many methods have been proposed to incorporate the user in the interestingness model. Some of these methods incorporate the background knowledge of the user to identify patterns that are surprising when contrasted to this background [59, 189], other methods require user interaction to learn her preferences and identify patterns that are interesting for her [72, 73, 220]. However, none of these works have proposed such an interestingness model for mining vertex-attributed graphs. Achieving this goal needs to find an answer for several questions: What kind of information can be considered as

background knowledge about vertex-attributes and graph structures? How to integrate both subjective constraints about attributes and edges into the same interestingness measure? To learn user preferences, how to interact with her? How to derive preferences from these interactions?

Another important challenge related to the interestingness measure is the redundancy issue. Patterns returned by mining methods can be very similar in the sense that they cover highly overlapping parts of the data, which gives redundant information. For example, algorithms for mining frequent itemsets can output a large number of patterns that cover almost the same subset of items. Analyzing such results is hard and overwhelming. One needs to design a data mining approach which is able to extract a diversified set of patterns that: (1) depicts as much information as possible about the dataset, (2) requires as few analyzing efforts as possible from the user. Moreover, it is very common that object attributes are related with a semantic hierarchy. For example, in the sale transactions dataset illustrated in Tab 1.1, an item i_1 ="skimmed milk" can be a a sub-type of another item i_3 ="milk". Informing the user that "skimmed milk" is highly frequent in the dataset would increase her expectation about the frequency of "milk". Then, informing her about the high frequency of "milk" becomes less interesting. How to consider this kind of dependency between attributes into the interestingness measure?

1.4 Contributions

This thesis addresses the problem of mining patterns in vertex-attributed graphs. We introduce new pattern languages, primitives and associated pattern discovery algorithms. The purpose is to improve the quality of patterns returned to the user, which requires the consideration of domain knowledge, user's prior knowledge, and user feedback. The proposed approaches can be structured into three main contributions:

Definition of Exceptional Attributed Subgraph Mining. This problem consists in identifying connected subgraph whose vertices share some characteristics that distinguish them from the rest of the graph. These characteristics are related to significantly high (or small) numerical values of some attributes in vertices of the exceptional subgraph. We define a data mining approach rooted in the Subgroup Discovery and Exceptional Model Mining frameworks. This contribution has been published in the IEEE ICDM 2016 conference proceedings [23], and then invited for an extended version in the journal of Knowledge and Information Systems [24].

Taking into account the user in the mining of vertex-attributed graphs. We design novel methods that account for the user in the interestingness model employed for evaluating patterns in vertex-attributed graphs. Two methods have been proposed:

- We design an approach that is built upon the Subjective Interestingness framework proposed by De Bie [59]. This method aims at incorporating user background knowledge, and accounts for the assimilation cost of a pattern, to identify patterns that are both informative (unexpected) and easy

to assimilate by the user. Also, we show how to update the interestingness model when a pattern is presented to the user, to continuously identify patterns that provide new information comparing to the already acquired knowledge from the previously presented patterns. This model updating technique is a principled strategy that aims to solve the redundancy issue in the results set. This work has been published in the workshop of Mining and Learning with Graphs 2018 [25], and an extended version has been accepted for publication in the Data Mining and Knowledge Discovery journal .

- We propose an approach that integrates user-preferences when mining attributed graphs. This method exploits an interactive process with the user to bias the pattern interestingness. It has been defined for the task of event detection in social media. In fact, event detection is a concrete case study where the incorporation of user preferences makes sense and can be evaluated with real users, as a large number of persons are able to interpret results of this task. This work has been accepted for publication in the IEEE Transactions on Knowledge and Data Engineering journal [27].

Integration of attributes dependencies into the interestingness model. In several datasets, a hierarchical relationship stands between attributes that describe objects (or vertices for graphs). In this case, information provided by different attributes is dependent and overlapping. We propose the first subjective interestingness model that takes advantage from a predefined attribute hierarchy to incorporate the information dependency between attributes. This work has been accepted for publication in the ACM SIGKDD 2019 conference proceedings [26].

1.5 Structure of the thesis

This thesis is organized as follows. Chapter 2 presents the state of the art of the related topics: (1) mining graphs and vertex-attributed graphs, (2) Subgroup Discovery and Exceptional Model Mining, (3) Incorporation of the user in the pattern discovery task. This chapter also discusses the limitations that we propose to address in this thesis. Chapter 3 presents our first contribution, which is the introduction of the problem of Exceptional Attributed Subgraph Mining. Chapter 4 presents a work related to the second contribution. We define a subjective interestingness model that integrates user's background knowledge when evaluating interestingness of attributed subgraphs. Chapter 5 introduces an attributed subgraph mining method that incorporates user's preferences through an interactive process. We show how this method has been used for the task of event detection in Twitter. In Chapter 6, we present a subjective interestingness that incorporates the attribute hierarchy into the background model. This model has been exploited to mine a particular subset of hierarchical attributes called antichains. Chapter 7 gives the conclusions and the perspectives of this thesis.

CHAPTER 2

STATE OF THE ART

Contents

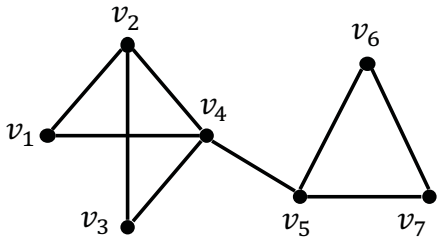


Figure 2.1: Example of a plain graph $G = (V, E)$ with the set of vertices $V = \{v_1, v_2, ..., v_7\}$.

Graphs are a powerful mathematical abstraction that can be used to represent a large variety of real world datasets where there are interactions between entities. Analyzing this kind of structure to extract useful knowledge is an important challenge for many scientific communities. Some of the applications that have been addressed using graphs are the analysis of communication networks [173], disease spreading [212], biological networks [17, 95, 104, 109], social networks [12, 93, 152, 224], World Wide Web [84, 91]. The simplest structure of graphs is $G = (V, E)$ (plain graphs), defined with a set of vertices V connected with a set of undirected edges $E \subseteq V \times V$. Fig 2.1 shows a toy graph $G = (V, E)$ with the set of vertices $V = \{v_1, v_2, v_3, ..., v_7\}$ and the set of edges $E = \{(v_1, v_2), (v_1, v_4), ...(v_6, v_7)\}$. As an example, a social network can be modelled with a graph whose vertices V represent users of the social network, and edges represent friendship relations between them (i.e., if the users corresponding to v_1 and v_2 are friends, then $(v_1, v_2) \in E$). Many data mining tasks have been defined on plain graphs, such as graph matching [186, 205], mining frequent subgraphs [117, 126, 222], mining cliques [11, 49, 89, 146], quasi-cliques [143, 169, 226], and pseudo-cliques [206], community detection [12, 44, 85, 224].

More complex forms of graphs have been defined and explored in the literature. In general terms, vertices and/or edges can be augmented with different kinds of attributes, and the graph structure can be made dynamic by introducing the temporal dimension. For example, in addition to vertices V and edges E, a vertex-attributed graph contains attributes that give information about the vertices. For a graph representing a social network with vertices corresponding to users, vertex attributes can provide information related to the users (e.g., age, topic of interest, profession, gender, etc.). In our thesis, we are interested in the problem of mining local patterns in vertex-attributed graphs. While many methods have been proposed to discover different kinds of patterns in these graphs [86, 98, 122, 156, 190], there is still a lot of challenges in this field. For instance, it is not easy to define a pattern syntax that allows to capture structures that are at the same time cohesive, interesting and easy to interpret. A pattern can be a subgraph, a subset of attributes, a property that describes all the graph or only a part of it. After coming up with a suitable pattern syntax, a difficulty is to define how to rank these patterns to determine the ones that are interesting. When talking about local patterns (e.g., subgraphs), these patterns are likely to be interesting when they are characterized by some properties that distinguish them from the overall graph, otherwise, these local patterns would be already expected. The task of discovering patterns that discriminate a subset of data from the overall dataset has been efficiently formalized with Subgroup Discovery [130] and Exceptional Model Mining [71]. Although these tasks have been initially

defined for simple datasets without graph structures, few recent works extend these concepts to mine graphs [15, 121]. In addition to the notion of pattern exceptionality regarding the whole graph, the pattern interestingness also depends on the final user, respectively, in her background knowledge and preferences. Several studies have been done to involve the user when assessing the quality of patterns. Some of these methods propose to consider the user background knowledge to find the patterns that are surprising when contrasted to these priors [59, 189], other methods require user interactions to learn her preferences and identify patterns that are interesting for her [72, 73, 220].

In this state of the art, we present previous works related to the aforementioned topics, that are: mining graphs, mining exceptional patterns, and taking into account the user in the mining process. In Section 2.1, we present data mining methods defined for different classes of graphs, then we detail particularly the works specific to vertex-attributed graphs in Section 2.2. In Section 2.3, we give a review about Subgroup Discovery and Exceptional Model Mining and their extension to graphs. In Section 2.4, we present methods that aim to involve the user when assessing the interestingness of data mining results.

2.1 Mining graphs

Several problems have been defined to extract knowledge from graphs, and many classes of graphs have been explored in the literature. In this section, we start by presenting works related to plain graphs $G = (V, E)$ with unweighted and undirected edges E, we then focus on more complex forms of graphs. First, let us define some notations that will be used next in this section. The neighborhood $N(v)$ of a vertex $v \in V$ is the set of vertices connected to v, $N(v) = \{u \mid (u, v) \in E\}$. The degree $deg(v)$ is the number of neighbors of v, $deg(v) = |N(v)|$. The subgraph $G[U]$ induced by a set of vertices $U \subseteq V$ is defined as G restricted to the vertices of U and containing only edges between vertices in U. The function $density(G[U])$ of the subgraph $G[U] = (U, E_U)$ is the number of edges in $G[U]$ divided by the maximum number of edges in a graph of size $|U|$, that is $density(G[U]) = \frac{|E_U|}{|U|\cdot(|U|-1)/2}$.

2.1.1 Plain graphs $G = (V, E)$

We now consider approaches defined for a plain graph $G = (V, E)$ as the one shown in Fig. 2.1. Several methods of this category aim to understand the global structure of the graph by computing macroscopic properties. Some of these properties are the degree distribution, the diameter, and the clustering coefficient. The degree distribution [13] is the probability distribution $\Pr_{deg}$ of the vertex degree over the whole graph. Formally, $\Pr_{deg}(k)$ is the probability of observing a vertex v with a degree $deg(v) = k$ in the studied graph. It has been shown that the degree distribution of many real world graphs follows a power law [18], i.e., $\Pr_{deg}(k) \backsim k^{-\gamma}$ (γ is a constant). Examples of these graphs are the world wide web and some social networks. Such graphs are called scale-free networks.

Another commonly used property to study graphs is the clustering coefficient [215]. This is a measure that quantifies to which extent vertices of the graph tend to cluster together. This coefficient is based on triplets of vertices. A triplet is three vertices that are connected by either two edges (open triplet) or

three edges (closed triplet). A triangle graph therefore includes three closed triplets, one centered on each of the vertices. The clustering coefficient C is defined as:

$$C = \frac{\text{number of closed triplets}}{\text{number of all triplets (open and closed)}}.$$

Another global property that has been largely studied for synthetic and real graphs is the diameter. Given that $d(u, v)$ is the distance (the shortest path) between the vertices u and v, the diameter $diam(G)$ of a graph G is the longest distance between two vertices of the graph: $diam(G) = \max_{u,v \in V} d(u, v)$. In Fig. 2.1, the diameter is 3 and it corresponds to the distance between v_1 and v_7. This measure can give an idea about whether or not vertices of the graph are close to each other. However, this measure is too sensitive to outliers. A more robust notion is the effective diameter, defined as the minimum distance for which a large fraction, typically 90%, of all connected pairs of vertices can reach each other.

Although these macroscopic properties can describe the general structure of the graph, they are not able to discover local patterns that can be related to only a part of the overall graph. Several problems have been defined to identify interesting local structures in plain graphs. To give a clear idea, we will explain some of the most familiar problems of this category: mining maximal cliques [49, 75, 89, 204] and maximal quasi-cliques [143, 169, 226].

Mining maximal cliques. A clique $U \subseteq V$ is a subset of vertices that are completely connected to each other, i.e., $\forall u, v \in U : (u, v) \in E$. A maximal clique U is a clique that is not a subset of another clique. In Fig. 2.1, $U_1 = \{v_5, v_6, v_7\}$ is a maximal clique, and $U_2 = \{v_5, v_6\}$ is a clique that is not maximal because $U_2 \subseteq U_1$. Mining maximal cliques can be used to identify communities from a graph, for example in a social network, a set of users that are all connected to each other. A clique is considered a local pattern, because its definition is related to only a subset of vertices $U \subseteq V$. The property of $U \subseteq V$ being a clique depends only on the connectivity of vertices of U, and it is completely independent of the rest of the graph. The problem of determining whether a graph contains a clique of at least a given size k is a NP-complete problem [119]. Many algorithms have been proposed to enumerate maximal cliques [75, 89, 204]. One of the most successful in practice is the Bron-Kerbosch algorithm [49], which uses a backtracking strategy that recursively solves subproblems derived from the main problem.

Mining maximal quasi-cliques. The enumeration of cliques imposes the hard constraint of full connectiveness. However in many cases, a lot of highly connected subgraphs are not cliques because only some edges are missing. Finding these structures can be also interesting and more practical for mining real world graphs. To this end, The notion of quasi-clique has been defined. Given a threshold $\delta \in (0, 1]$, a δ-quasi-clique $U \subseteq V$ is a subset of vertices such that:

$$\forall v \in U : \frac{|N(v) \cap U|}{|U| - 1} \geq \delta.$$

This definition allows a bounded number of missing edges for each vertex. In Fig. 2.1, $U = \{v_1, v_2, v_3, v_4\}$ is a $\frac{2}{3}$-quasi-clique, because each $v \in U$ is connected to at least 2 out of 3 vertices. A δ-quasi-clique is

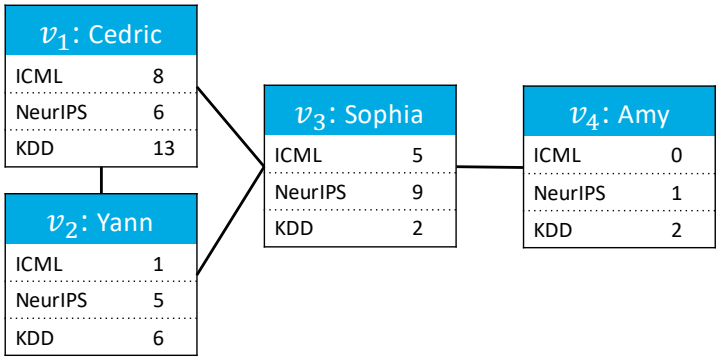


Figure 2.2: Toy example of vertex-attributed graph: DBLP co-authorship network.

maximal if it is not included in another δ-quasi-clique. Many algorithms have been proposed to mine maximal quasi-cliques. Some of these methods exhaustively enumerate the complete set of quasi-cliques by using some pruning techniques to optimize the search space exploration [143, 169, 226]. Other methods use heuristic or randomized methods [3, 50, 149], although they do not provide the complete results set, they can be more efficient than exhaustive approaches. In the literature, we also find the notion of pseudo-clique which is defined based on a minimal density threshold applied on a subgraph as a whole [206]. Formally, $U \subseteq V$ is a δ-pseudo clique if $density(U) \geq \delta$. Whilst the formal problem definitions of mining quasi-cliques and pseudo-cliques are different, they both aim to identify local subgraphs that are highly connected.

2.1.2 Augmented graphs

In many datasets, the simple form of graphs $G = (V, E)$ can be enriched by additional information about vertices, edges, the graph structure, etc. This leads to the definition of more complex classes of graphs, whose mining allows to provide more insights about the studied dataset, but poses additional challenges. In what follows, we present each of vertex-attributed graphs, edge-attributed graphs, and dynamic graphs, and we give examples of problems and applications that have been done based on these structures.

Vertex-attributed graphs. In various case studies, some additional descriptions are available about the entities corresponding to vertices. A toy vertex-attributed graph is shown in Fig. 2.2 (the DBLP co-authorship network). Each author is represented by a vertex. Each of them is described by 3 attributes, which are the number of publications in the following conferences: ICML, NeurIPS, KDD. An edge links two researchers who co-authored a paper. There has been a significant interest on studying graphs where vertices are described by attributes [44, 98, 159]. These attributes can be ordinal or not (e.g., numerical, categorical, Boolean). Many problems have been defined to find global or local patterns present in these structures, such as finding dense subgraphs with similar attributes, finding outlier vertices, etc. A more detailed review of these problems is given in Section 2.2.

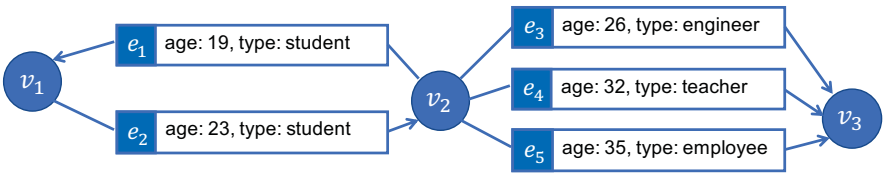


Figure 2.3: Toy example of edge-attributed graph: bike trips between three bike stations $\{v_1, v_2, v_3\}$. Each edge (trip) is described by the age and the profession of the bike user.

Edge-attributed graphs. In several datasets, we have additional information about edges. For instance, consider a graph representing bike trips in a city: vertices are bike stations and each edge corresponds to a bike trip from a station to another. Each edge can be described by information about the person using the bike such as age and profession. This can be well represented by edge-attributed graphs, i.e., graphs with attributes on edges. We show in Fig. 2.3 a toy edge-attributed graph representing bike trips between three stations $\{v_1, v_2, v_3\}$. Some approaches use edge information to define a similarity measure on edges in order to define dense subgraphs or communities [28, 40, 176]. In [121], we use edge-attributes to determine contexts for which there is connected subgraphs having exceptionally prevalent number of edges. For example, this approach can be used to discover a subgraph of bike stations having a high number of trips done by student whose age is between 15 and 20. Some methods have been defined to mine multilayer networks, i.e., graphs in which there is different categories of edges [34], this can be seen as an edge-attributed graph with one categorical attribute.

Dynamic graphs. A graph is dynamic if its structure evolves over time. Formally, it can be defined as a sequence $\mathcal{G} = \{G_1, G_2, ..., G_n\}$ over timestamps $T = \{t_1, ..., t_n\}$ of graphs $G_t = (V, E_t)$ with a constant set of vertices V and a set of edges $E_t \subseteq V \times V$ depending on time. In Fig. 2.4, a dynamic graph defined on three timestamps is shown. A dynamic graph can also be augmented by attributes in vertices or edges, and attribute-values can also be dynamic. Numerous approaches propose to mine dynamic graphs. Borgwardt et al. [41] introduce the problem of mining frequent subgraphs in dynamic graphs, i.e. identical graphs that appear in consecutive timestamps. Robardet [178] proposes an algorithm to mine pseudo-cliques which appear in consecutive timestamps with slight evolutions. Ahmed and Karypis [8] mine the evolution of conserved relational states, i.e. sequences of time-conserved patterns on consecutive time. Desmier et al. [64] define a new pattern domain that relies on the graph structure and the temporal evolution of the attribute values. It allows one to discover subgraphs of small diameter whose vertex attributes follow the same trends.

2.2 Mining vertex-attributed graphs

In this section, we review more extensively the literature of vertex-attributed graphs, as the methods proposed in this thesis are mainly based on this particular structure. Let us first give a formal definition

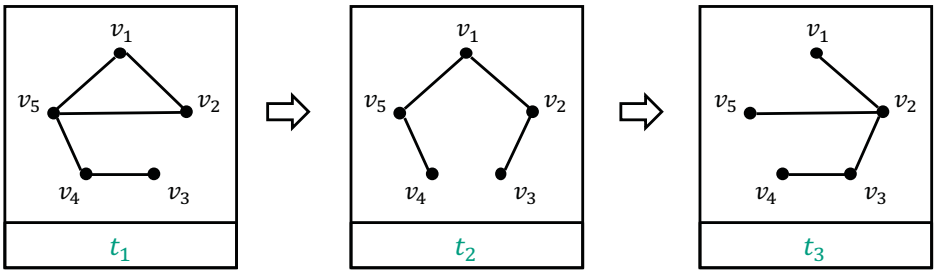


Figure 2.4: Toy example of dynamic graph defined for three timestamps $T = \{t_1, t_2, t_3\}$.

to vertex-attributed graphs.

Definition 2.1. A vertex-attributed graph $G = (V, E, \hat{A})$ is defined by a set of vertices V, a set of edges $E \subseteq V \times V$, and a set of attributes $\hat{A}$ over vertices (formally, functions mapping a vertex onto an attribute value), with $\hat{a}(v) \in Dom_a$ denoting the value of attribute $\hat{a} \in \hat{A}$ on $v \in V$.

We use hats in $\hat{a}$ and $\hat{A}$ to signify the empirical values of the attributes observed on the graph G. The notations a and A represent the set of attributes in a more generic way, they are used to denote (possibly random) variables over the same domains, and to denote the pattern syntax.

In the following, we start by presenting community detection methods in vertex-attributed graphs. The goal of these methods is to group vertices into subgraphs that are dense and share similar attribute values. These are usually considered as global approaches, since they aim to find a partition of the data that optimizes a goodness function which is defined on the whole graph. Then, we present methods that identify local patterns in vertex-attributed graphs. While community detection methods optimize a global measure, local pattern mining methods identify patterns that can be present in only a part of the graph. We then present the outlier detection in vertex-attributed graphs. This problem aims to identify vertices that are somehow anomalous regarding their connectivity or their attribute values. Finally, we describe the community search problem, where the purpose is to identify a community that contains a user-specified set of vertices.

2.2.1 Community detection and clustering in vertex-attributed graphs

Clustering and Community detection algorithms partition a dataset into groups of similar and/or interacting objects. Traditional community detection algorithms perform on simple graphs without attributes, and they aim to find communities by focusing only on the network structure. For some methods, these graphs can be weighted, i.e., graphs $G = (V, E, w)$ with a weight function $w : E \longrightarrow [0, 1]$ that assigns a value to each edge. The weight $w(u, v)$ of an edge $(u, v) \in E$ represents how much the connection between u and v is strong. For this kind of graphs, standard community detection methods partition the vertices into dense subgraphs where edges have high values of weight. Several recent papers proposed new community detection methods that consider both network connections and vertex attributes. Some methods proposed to first transform the attributed graph to a single weighted graph, where weights

represent attribute similarity [161, 230]. Then, any clustering algorithm for weighted graphs can be used. Numerous existing methods combine network and vertex attribute information into the same model [194, 221]. In Fig. 2.5, we take from Bojchevski and Günnemann [35] an example of attributed graph with three Boolean attributes. This graph can be partitionned into the three communities shown with different colors. The vertices of each of community are highly connected with each other and weakly connected with vertices outside their community, and they also have similar Boolean attribute values.

Formally, a clustering can be seen as a function $C : V \longrightarrow \{1, ..., f\}$ that assigns each vertex $v \in V$ to a cluster (a community) $i \in \{1, ..., f\}$, the total number of communities is $f \in [\![1, |V|]\!]$. There is a large number of possible partitionnings of the data, let us denote them by $\mathcal{C}$, and $C \in \mathcal{C}$ is one possible clustering. The quality of a clustering C of a graph G is generally measured by some objective function $F(G, C)$. The clustering task of G can be defined as finding the partitionning $C \in \mathcal{C}$ that optimizes the following criteria: $C = \text{argmax}_{C' \in \mathcal{C}} F(G, C')$. This definition allows one to assign each vertex to only a single cluster. While many of community detection methods in attributed graphs use single-assignment clustering for each vertex [12, 76, 180, 231], Yang et al. [224] developed an algorithm that identifies overlapping communities using a statistical model of interaction between network structure and vertex attributes. Different models have been used to assess the quality of a partitionning, and several algorithms have been defined to optimize these models. An extensive survey of these works is presented in [44].

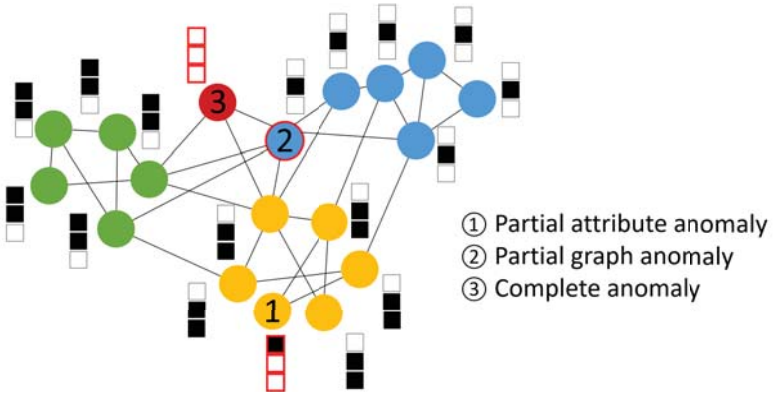


Figure 2.5: Example of clustering and outlier detection in an attributed graph, taken from [35]. Three detected communities (with different colors), and three detected outliers.

2.2.2 Local pattern discovery in vertex-attributed graphs

Many approaches have been designed to discover local patterns in such structures. While community detection methods are global approaches whose results depict the whole graph, local pattern mining methods identify patterns that uncover substructures of the graph (e.g., a subset of vertices or attributes). This results in the definition of new classes of patterns: cohesive patterns, subspace clustering, proximity patterns, itemset-sharing subgraph set, structural correlation patterns, etc.

Subspace clustering. The goal of this problem is to identify dense subgraphs whose vertices have similar values of some attributes. A difference with standard community detection is that, in subspace clustering, the attributes-similarity between vertices of the same cluster is assessed regarding only a subspace of attributes. Indeed, some dimensions are ofen noisy and not relevant for all clusters. Also, subspace clustering has been proposed as a way to deal with the curse of dimensionality [29] when the number of attributes is huge. There is several formal definitions of this task, as an example, let us detail the one proposed in [98] for graphs with numerical vertex-attributes.

Definition 2.2 (Subspace clustering in attributed graphs). Given a graph $G = (V, E, \hat{A})$ with numerical attributes, a threshold s_{min} (resp. u_{min}) on the minimum number of attributes (resp. vertices), a threshold w on similarity between attributes, and a threshold $\delta \in [\![0, 1]\!]$ on the minimum density, a subspace cluster (U, S) is defined with a set of vertices $U \subseteq V$ and a set of attributes $S \subseteq A$ such that:

- $|U| \geq u_{min}$ and $|S| \geq s_{min}$,
- the density of $G[U]$, the subgraph induced by U, is higher than δ,
- $\forall a \in S, \forall u, v \in U : |\hat{a}(u) - \hat{a}(v)| \leq w$,
- $\forall a \in A \setminus S, \exists u, v \in U : |\hat{a}(u) - \hat{a}(v)| > w$.

Many other papers have addressed this problem and have proposed different solutions. In [99, 100], a method is designed to identify subspace clusters with arbitrary shape and size, using a density-based cluster definition. Another similar method based on spectral analysis is defined in [101].

Cohesive patterns. Moser et al. [156] propose one of the first approaches that consider simultanuously graph structure and vertex-attributes in the dense subgraph mining problem. This method is defined particularly for graphs with categorical attributes. They define the cohesive pattern as a subgraph that is (1) induced by a connected δ-pseudo-clique, and (2) homogenous regarding a subspace of attributes. We recall that a subset of vertices $U \subseteq V$ is a δ-pseudo-clique if $density(G[U]) \geq \delta$. In order to evaluate the homogeneity of $U \subseteq V$ in a subspace of attributes $B \subseteq A$, a Boolean subspace cohesion function $s(U, B, \theta_s)$ is defined and used. This function returns True if the homogeneity of U in B is higher than a threshold θ_s. The framework proposed in [156] is somehow flexible, as the function computing the homogeneity value is not imposed but it only needs to be anti-monotonic w.r.t U and B, i.e., if $U \subseteq U' \subseteq V$ and $B \subseteq B' \subseteq A$, then $s(U', B', \theta_s) \implies s(U, B, \theta_s)$. Moser et al. [156] propose an algorithm to extract cohesive patterns that are maximal (i.e., not included in larger cohesive patterns).

Proximity patterns. Khan et al. [122] define the problem of mining proximity patterns in graphs where vertices are described by Boolean attributes (called items). A proximity pattern is a set of attributes that repeatedly appear in multiple tightly connected subgraphs. This problem is related to the frequent itemset mining task. However, the itemset does not necessarily need to appear in each vertex to be accounted, but on neighbor vertices. This makes the pattern definition more flexible and able to capture better the

information that is present in the studied graph. We take from [122] an example shown in Fig. 2.6. It illustrates a proximity pattern $\{a, b, c\}$ in a graph whose vertices are described by the Boolean attributes $A = \{a, b, c, d, e, f\}$. The items $\{a, b, c\}$ often occur together, but it is neither a frequent subgraph, nor a frequent itemset if we require each vertex to have all of the three items $\{a, b, c\}$. The specificity of proximity patterns is their flexibility. In order to mine these patterns, Khan et al. [122] propose two models: the neighbor association model and the information propagation model. The first one implies an exhaustive strategy that requires to generate all the subgraphs that embed a pattern, i.e., all subsets of vertices that contain the pattern. The authors experimentally show that this model is not tractable in practice. The second model is based on the idea that considers the graph as a representation of reality at a given timestamp. The information (attributes) is propagated through the graph until the graph reaches a stable state $\tilde{G}$. Then, the proximity patterns are mined as frequent itemsets in $\tilde{G}$.

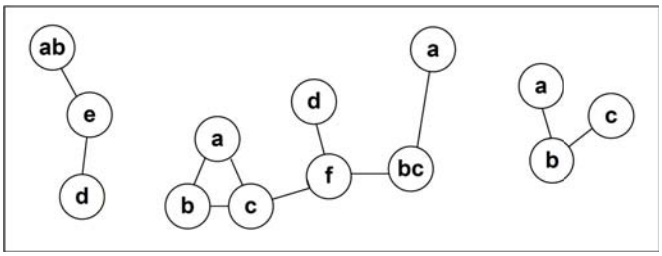


Figure 2.6: Proximity pattern $\{a, b, c\}$, an example taken from [122].

Itemset-sharing subgraph set. This problem has been proposed in [86, 187] for graphs with Boolean vertex-attributes. An itemset-sharing subgraph set is defined as a collection of connected components in the subgraph induced by a non-empty set of attributes, with a minimum bound on the number of vertices in the connected components. The subgraph induced by an itemset $S \subseteq A$ is constituted of vertices that contain each of the items in S. This problem does not required the itemset-sharing subgraph to be dense, they are only required to be connected.

Structural correlation patterns. This problem has been proposed in [190, 191] where the authors study the correlation between Boolean vertex-attributes and dense subgraphs. A set of attributes $S \subseteq A$ is structurally correlated if, in the subgraph induced by S, a large percentage of vertices are in quasi-cliques. In other terms, the structural correlation measure of a set of attributes S is the ratio between the number of vertices belonging to quasi-cliques in the subgraph induced by S and the number of total vertices in this induced subgraph. Then, a structural correlated pattern (U, S) is defined by a structurally correlated set of attributes $S \subseteq A$, and a quasi-cliques $U \subseteq V$ that belongs to the induced subgraph by S. The solution proposed in [190, 191] also allows to specify thresholds on the size of the attributes set S, and the size of the quasi-clique U. An algorithm is then proposed to enumerate these patterns using a depth-first search manner that employs pruning techniques to avoid exploring some of the non promising parts of the search space.

K-clique percolated components. This class of pattern has been defined in [157]. The authors consider a graph where Boolean attributes are associated to vertices, and the goal is to identify components made of overlapping homogeneous cliques. The homogeneity means sharing a common set of attributes having the same value. Given that $F(v) = \{a \in A \mid \hat{a}(v) = true\}$, a collection of set of vertices $M = \{U_1, U_2, ...\}$ is a k-clique percolated component if: (1) The number of attributes shared by all sets of vertices is greater than a threshold α, i.e., $ATB(M) = \bigcap_{U \in M} \bigcap_{v \in U} F(v) > \alpha$, (2) M contains at least δ cliques of size k, and these cliques are connected by overlaps of $k-1$ vertices, (3) M contains all the k-cliques that share the attributes $ATB(M)$. Mougel et al. [157] also propose a complete algorithm to mine k-clique percolated components using a subgraph enumeration strategy.

Topological patterns. Prado et al. [175] propose to mine the topology of attributed graphs by finding regularities among ordinal vertex descriptors. Two types of descriptors are considered: (1) the vertex-attributes that are initially given with the input graph, (2) some topological properties used to describe the connectivity of the vertices such as the degree, the clustering coefficient, the betweenness centrality, etc. A topological pattern is defined as a set of descriptors associated to a trend. For example, given that $\{a, b, c\}$ are vertex descriptors, a topological pattern $\{a^+, b^+, c^-\}$ can be read: the more a, the more b, and the less c in vertices of G. If we consider the DBLP co-authorship graph where vertices represent authors and are described by their h-index, and the number of hours spent on instructional duties, a pattern could represent a set of authors that have a high h-index, spend low time in instructional duties and have a high betweenness centrality (they publish with co-authors that are central in the graph). Prado et al. [175] define several interestingness measures for topological patterns, and propose an algorithm to extract such patterns.

Several other tasks have also been defined for mining local patterns. Mougel et al. [158] identifies large cliques where all vertices share enough Boolean attributes. Zhang et al. [229] aim to find (k, r)-cores in attributed graphs. A (k, r)-cores is a connected subgraph where each vertex is connected to at least k vertices and the similarity between attributes values is higher than a minimum threshold r. Silva et al. [192] propose an approach to compress attributes values in a graph where each vertex is described by a single numerical attribute. This consists in partitioning the graph into smooth areas where attribute values are similar, and store only the average values of each area.

2.2.3 Outlier detection in vertex-attributed graphs

Outlier detection is an important problem that has proven its high utility through a tremendous number of applications. It aims to detect objects and structures that are unexpected, anomalous, rare and suspicious in a dataset. One of its research subfields is outlier detection in attributed graphs where the goal is generally to detect anomalous vertices [35, 136, 137, 159, 171, 174, 185]. The anomaly of a vertex in an attributed graph can be related to its connectivity (graph structure), or to its attribute values, or to both of them. For example, a mixture model is introduced in [137] to identify and describe anomalous vertices in attributed graphs by integrating both structural and attribute information into the probabilistic

model. Some of anomaly detection methods are binary, i.e., each v vertex is either an outlier or not. Other methods assign an "outlier score" that tells to which extent a vertex v is an outlier, and vertices can then be ranked based on this score.

Measuring the deviation of vertices may be ambiguous and noisy when considering simultanuously graph structure and all attributes. In fact, some attributes may be irrelevant, but also, the notion of anomaly of a vertex can be relative to its local community. To solve this issue, several approaches assess the anomaly score of each vertex regarding to a subspace cluster to which it belongs (a subset of dense vertices that are similar w.r.t. a subset of attributes) [159, 171, 185]. In some papers [35, 136], outlier vertices are categorized into several types defined based on whether the anomaly is caused by attribute or structure deviation or both of them. Defining anomaly types can make it easier to identify them. In [35], the authors derive three types of anomalies shown in Fig. 2.5, (1) partial attribute anomaly: Although vertex 1 structurally belongs to the yellow community, it has completely different attributes compared to the other vertices in this community, (2) partial graph anomaly: vertex 2 has exactly the same attribute values as those of vertices in the blue community, however it does not fit in this community regarding to the network structure as it is connected to several vertices from different communities, (3) complete anomaly: vertex 3 does not fit in any community from neither a structure aspect nor attribute values.

2.2.4 Community search in vertex-attributed graphs

While the goal of community detection methods is to partition the graph into groups of strongly interacting vertices, the community search problem aims at identifying the community to which a specified set of vertices $Q \subseteq V$ belongs. The community search problem can be tricky to handle with only using community detection methods. As motivated in [197] and illustrated in Fig. 2.7, if Bob and Alice take the same tango class, and Charles is Bob's boss, the community formed by Bob and Alice is very different than the community formed by Bob and Charles. Sozio and Gionis [197] define the community search problem for plain graphs as follows.

Definition 2.3. Given a graph $G = (V, E)$, a set of query vertices $Q \subseteq V$, and a community goodness function f, the goal is to find an induced subgraph $H = (V_H, E_H)$ of G such that: (1) $Q \subseteq V_H$, (2) H is connected, and (3) $f(H)$ is maximized among all feasible choices for H.

Numerous works extended this problem to attributed graphs [78–82, 113, 114, 188], where the identified community containing the query vertices also needs to satisfy attributes cohesiveness. Fang et al. [81] integrate the spatial information for the vertices in the community search problem in attributed graphs, this allows to search communities situated in a close spatial area. Huang and Lakshmanan [113] introduces the problem of attribute-driven community search, where the user can also specify a set of query attributes that need to be similar in the identified community. Fang et al. [78] extends the problem of community search to graphs whose vertex-attributes are arranged in a hierarchical manner, and designed a method that is able to choose the appropriate level of hierarchy for similar attributes of the community.

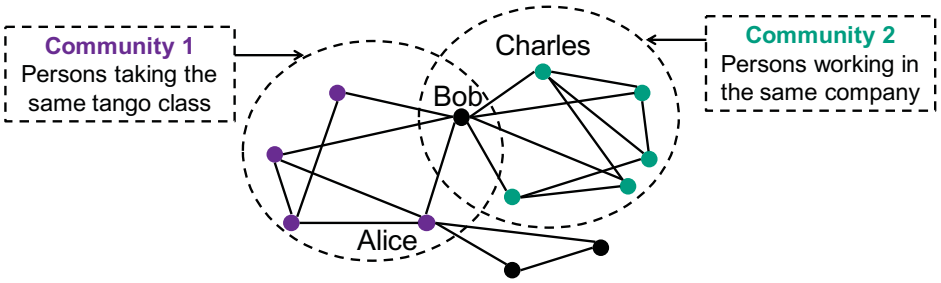


Figure 2.7: Example for the community search problem. Results for this task would be the community 1 if the query vertices are {Bob, Alice}, and the community 2 if the query vertices are {Bob, Charles}.

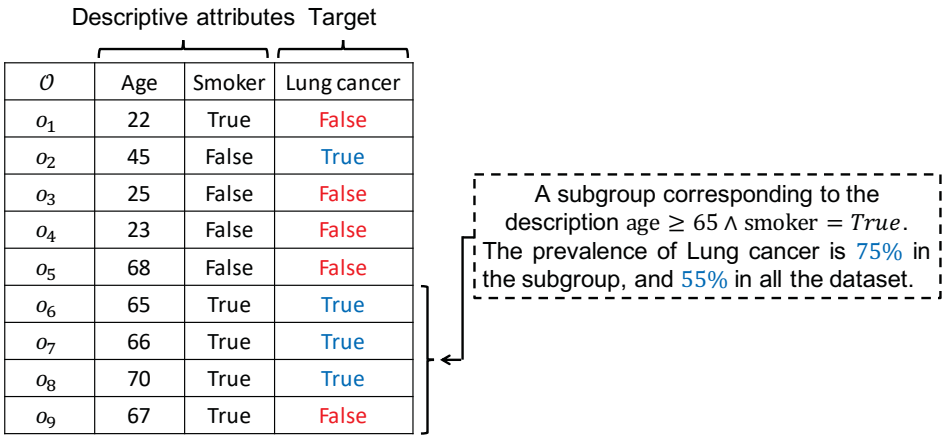
2.3 Supervised descriptive rule discovery

Data mining and machine learning community has shown a great interest in the task of discovering patterns that discriminate a subset of data from the overall dataset. Such task can be efficiently formalized in Subgroup Discovery (SD) [125, 130, 164, 219] and Exceptional Model Mining (EMM) [71, 133].

2.3.1 Subgroup Discovery (SD)

This task aims to find descriptions of sub-populations for which the distribution of a predefined target value is significantly different from the distribution in the whole data. In Fig. 2.8, we show a toy dataset containing objects describing 10 patients, each of them has two descriptive attributes: age (numerical), $smoker$ (Boolean), and the target attribute $lung\_cancer$ (Boolean). A subgroup defined by the description $age \geq 65 \wedge smoker = true$ fosters the prevalence of the target variable $lung\_cancer$ in this dataset. A subgroup is a description generalization whose discriminating ability is assessed by a quality measure. While the most commonly used quality measures to assess the interest of a subgroup is the Weighted Relative Accuracy ($WRAcc$) [128], a large variety of measures exists in the literature: the F-Score, the Jaccard coefficients, the Weighted Kullback Leibler divergence (WKL), the Subjective Interestingness (SI), etc. In addition to SD, two other close notions have been independently formalized and then unified by Novak [164]: Contrast Set Mining [19] and emerging pattern mining [66]. Many other similar data mining tasks have been defined, for instance, significant rules [199], classification association rule [142]. However, SD can be considered as the most generic task since it has been defined as a framework that is agnostic of the data and the pattern domain.

The main challenges that have been studied for Subgroup Discovery are: (1) The pattern language: what are the constraints/restrictions of attributes that can be used to select a subgroup from the dataset? (2) The quality measure: how to assess the interestingness of a subgroup? (3) Algorithms: how to explore the search space to identify subgroups of interest? Numerous exhaustive and heuristic algorithms have been proposed in the literature. The first exploration methods that have been proposed for SD implemented an exhaustive search, i.e., the search space is completely explored ensuring that the best subgroups are found. Some of the exhaustive approaches that use efficient pruning strategies to explore the search space are SD-



$\mathcal{O}$	Age	Smoker	Lung cancer
o_1	22	True	False
o_2	45	False	True
o_3	25	False	False
o_4	23	False	False
o_5	68	False	False
o_6	65	True	True
o_7	66	True	True
o_8	70	True	True
o_9	67	True	False

Figure 2.8: Example for Subgroup Discovery. A toy dataset of objects having two descriptive attributes and a target attribute.

MAP [16] and BSD [134]. Because exhaustive approaches have scalability issues, heuristic algorithms have been also proposed. The commonly used heuristic strategy is the beam search [87, 129]. Some other heuristic schemes have also been used such as evolutionary algorithms [62, 165, 179] and output sampling approaches [37, 38, 153]. Interestingly, some recent papers propose anytime algorithm [22, 43], i.e., (1) they yeld progressively subgroups whose quality and diversity improves over time, (2) they can be interrupted anytime and they have the guarantee to return the best subgroups if enough time is provided. Particularly, Belfodil et al. [22] provide a guarantee bounding the error on the top subgroup quality when the algorithm is interrupted before finding the best subgroup.

2.3.2 Exceptional Model Mining (EMM)

Usually considered as a generalization of SD, Exceptional Model Mining [71, 133] allows one to handle more complex target concepts, and use more sophisticated model to assess the deviation of subgroups. The main measures and algorithms of standard SD are generally defined for datasets where there is only one target variable. In EMM, there is no longer only one target variables but several, and a model over these targets is chosen to be the target concept. EMM aims to find subgroups of objects for which the model induced over the target variables significantly deviates from the model induced by all the dataset. We show in Fig. 2.9 an illustration of this framework proposed in [42]. Numerous EMM models defined on a set of target variables have been proposed: regression model [70] and correlation model [71] between two target variables, classification model as a target [68], preferences among a set of targets [61], Bayesian networks [69], etc. While the models used in EMM are more complex, the same algorithmic strategies as SD are used to identify exceptional subgroups, in fact, the search space of subgroups in EMM remain similar the ones of SD.

Example for EMM regression model. A concrete example of EMM is given in [70] where the goal is to study the relation between price and demand in economy. The economic law of demand states

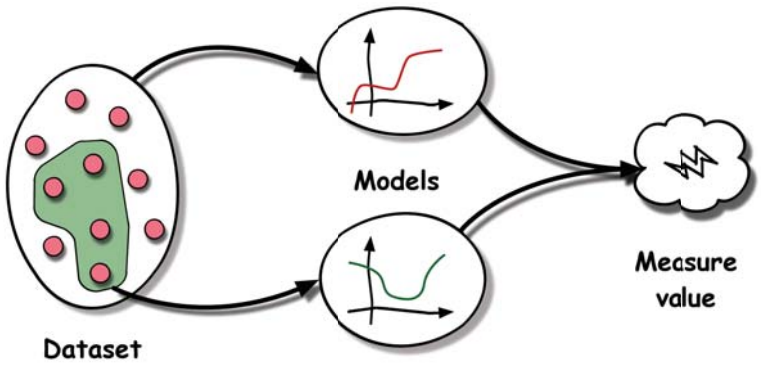
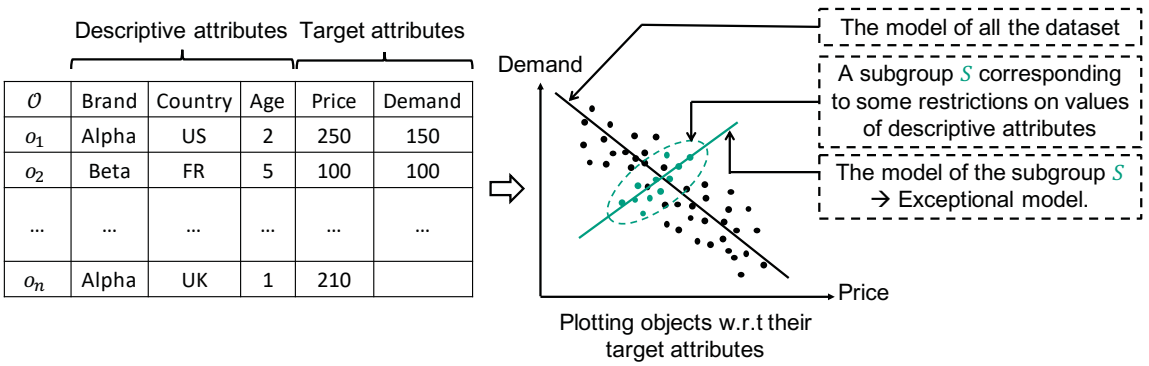


Figure 2.9: A typical EMM instance [42].

that (all else equal) if the price of a product increases, the demand for the product will decrease. In a regression model this would result in a negative slope when we regress demand on price. However, under specific conditions, people tend to buy more of a product when the price increases [116]. Hence, for those exceptional cases, we would get a positive slope of the regression line. Those exceptional cases can be investigated using EMM applied to a dataset of objects corresponding to products. Each product has descriptive attributes (e.g., category, age, brand), and two target variables: price and demand. A regression model between price and demand is used in [70]. Applying EMM allows one to find restrictions of attributes that identify subgroups having a regression model of target variables significantly different from the global regression model of targets over all the dataset. We illustrate this example in Fig. 2.10.



$\mathcal{O}$	Brand	Country	Age	Price	Demand
o_1	Alpha	US	2	250	150
o_2	Beta	FR	5	100	100
...	...	...	...	...	...
o_n	Alpha	UK	1	210	

Figure 2.10: Example of EMM with a regression model on two target attributes.

2.3.3 Exploiting SD/EMM for mining graphs

Although SD and EMM are initially defined for simple datasets of objects, recent works show interests in extending these concepts for attributed graphs. Atzmueller et al. [15] propose to mine descriptions of communities from vertex attributes, with an SD approach. In this supervised setting, each community is treated as a target that can be assessed by well-established measures, like the WRAcc measure. In [121],

we proposed a method for edge-attributed graphs to discover contextualized subgraphs that are exceptional with respect to a model of the dataset. Restrictions on the attributes, that are associated to edges, are used to generate subgraphs. Such patterns are of interest if they pass a statistical test and have high value on an adapted WRAcc measure. Similarly, Lemmerich et al. [135] propose to discover subgroups with exceptional transition behavior as assessed by a first-order Markov chain model. Some other works from Signal processing community also aim to discover subgraphs having some exceptionalities. For instance, Miller et al. [151] studied the problem of subgraph detection (SPG) in simple graphs with weighted edges, and formalised it as a problem of detecting signal in noise. It aims to find subgraph having an exceptional connectivity regarding to the one expected using a background model.

2.4 Taking into accout the user in the mining process

The goal of data mining is the discovery of interpretable patterns that ideally are interesting for the user. In practice, the notion of pattern interestingness strongly depends on the user, respectively, on her prior knowledge or her preferences. For instance, a pattern P may be interesting for a user u_1 but not for another user u_2, because u_2 already expected the presence of P in the data, or because P is related to a topic that does not interest u_2. For this reason, several works proposed interestingness models that consider not only the data but also the user to assess the quality of a pattern, which makes this quality a subjective measure. The problem of taking subjective interestingness into account in pattern mining was already identified in Silberschatz and Tuzhilin [189] and has seen a renewed interest in the last decade. Subjective models can be divided into two categories: (1) models that incorporate the user prior knowledge, and (2) models that integrate the user preferences.

2.4.1 Incorporation of user background knowledge

These methods aim to integrate the background knowledge of the user into the interestingness model in order to discover patterns that are surprising regarding these prior beliefs. Among them, the prominent work of De Bie [59] defines a general and statistically-founded framework that uses concepts from Information Theory. The user's state of mind is modeled as constraints on a probability distribution, called the background distribution, which represents the uncertainty the user has about the data. More specifically, the prior information is represented by the maximum entropy (MaxEnt) distribution subject to these constraints. This model is then used to capture patterns that are surprising regarding user beliefs. But also, this framework proposes to favor patterns that are easy to assimilate by the user. The assimilation cost of a pattern by the user is measured using a description length measure, which gives the number of bits required to encode the pattern in some specific encoding scheme. The subjective interestingness $SI(P)$ of a pattern P is the ratio between $IC(P)$ the information content and $DL(P)$ the description length of the pattern, thus representing the information density within the pattern. Also, when a pattern is displayed to the user, the background model is updated to integrate this pattern as already known by the user. This allows the approach to continuously return patterns that contain new piece of information

comparing to the already extracted ones. An illustration of this framework is given in Fig. 2.11. This framework has been successfully exploited in several data mining tasks. For example, De Bie [60] defined a subjective interestingness to assess the quality of tiles in a binary dataset where row sums and column sums are assumed to be priors. Lijffijt et al. [141] also used this framework for the problem of pattern mining in multi-relational databases. The user is considered to have prior beliefs on the degree of each entity in the different relationship types. In [208], the authors showed how this subjective interestingness can be exploited to capture surprisingly dense subgraphs in plain graphs. In this work, they consider two cases of background knowledge: (1) when the user only has beliefs about the overal density of the graph, and (2) when the user has prior beliefs about the degree of vertices. Kang et al. [118] applied this framework for the dimensionality reduction problem to seek for the most informative low dimensional representation. Several types of priors have been studied in this work, e.g., scale of the data (average variances), magnitude of spread, pairwise data similarities of some data points. Similarly, Adriaens et al. [4] designed a method to mine subjectively interesting trees connecting a set of query vertices in a graph. They showed how to derive the interestingness model for two types of priors: (1) representation of some partial or total order of vertices as priors (e.g., when vertices corresponding to events in time), (2) modeling knowledge about degree assortativity, i.e., when vertices have the tendency to connect to vertices of similar degree.

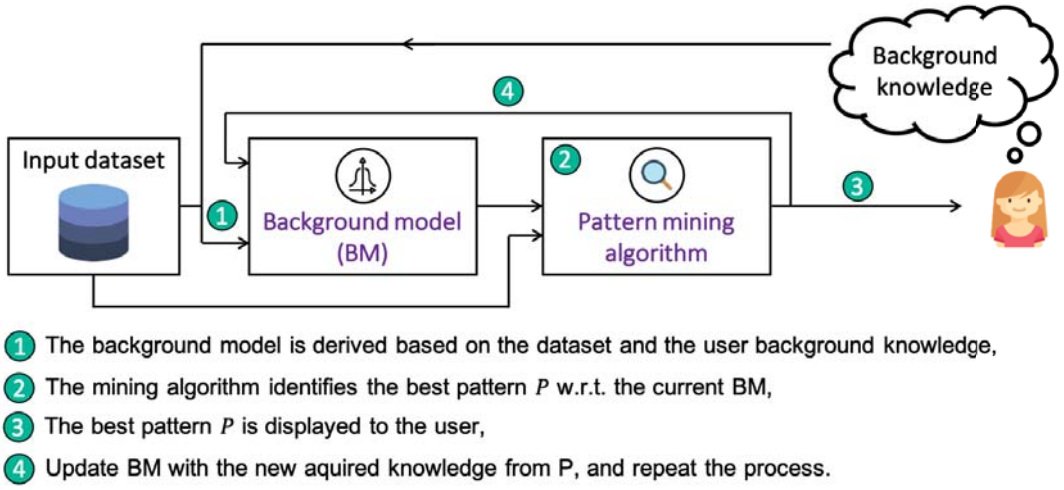


Figure 2.11: Overview of the subjective interestingness framework proposed in [59].

Subjective interestingness for mining tiles in binary databases. In order to give a clearer idea about the subjective interestingness proposed by De Bie [59], we explain in more details the model that has been used to assess the interestingness of tiles in binary databases [60]. A matrix $\hat{D}$ is used to denote a database with m rows and n columns. $\hat{D}(i,j) \in \{0,1\}$ is a Boolean cell located in the i-th row and j-th column of D. Patterns of interest are tiles in this study. A tile $\tau = (I,J)$ is defined as an ordered pair of a set of rows $I \subseteq \{1,...,m\}$ and a set of columns $J = \{1,...,n\}$. We say that a tile $\tau = (I,J)$

is present in the database $\hat{D}$, denoted $\tau \in \hat{D}$, iff $\hat{D}(i,j) = 1$ for all $i \in I$ and $j \in J$. The goal of this problem is to identify tiles that are subjectively interesting when contrasted to a background model. More precisely, the purpose is to identify tiles τ with the highest values of $\text{SI}(\tau) = \frac{\text{IC}(\tau)}{\text{DL}(\tau)}$ which is the ratio between the informativeness $\text{IC}(\tau)$ and the assimilation cost $\text{DL}(\tau)$. The information that are considered as background knowledge in this model are the sum of each row and the sum of each column. Any pattern that can be explained by referring to row or column sums in a binary database is then deemed uninformative. This hypothesis has been considered before, using computation intensive approaches based on swap randomization [92]. The MaxEnt model employed in [60] can be computed and used more efficiently in this situation for the same purpose as swap randomizations. While the user is considered to expect the right values for the row and column sums, she does not know the exact values of cells in $\hat{D}$. The uncertainty of the user about $\hat{D}$ is represented by the random matrix D that takes its values in $\{0,1\}^{m \times n}$. We need to infer a model that allows to compute the probability $\Pr(\tau \in D)$ conditionned to the background knowledge, which allows to evaluate how much a tile τ is surprising. The prior knowledge about the row and column sums are modeled as constraints on the probability distribution $\Pr(\tau \in D)$:

1. **The row sums are priors:** For each $i \in \{1, ..., m\}$, the users's expectation about the sum of the i-th row is equal to its empirical sum in $\hat{D}$:

$$\sum_{D \in \{0,1\}^{m \times n}} \left(\Pr(D) \cdot \sum_{j \in 1}^{n} D(i,j) \right) = \sum_{j \in 1}^{n} \hat{D}(i,j)$$

2. **The column sums are priors:** For each $j \in \{1, ..., n\}$, the users's expectation about the sum of the j-th column is equal to its empirical sum in $\hat{D}$:

$$\sum_{D \in \{0,1\}^{m \times n}} \left(\Pr(D) \cdot \sum_{i \in 1}^{m} D(i,j) \right) = \sum_{i \in 1}^{m} \hat{D}(i,j)$$

These constraints will not be sufficient to uniquely determine a probability distribution. The solution proposed in [60] is to search for the MaxEnt distribution, i.e., the probability distribution that has the largest entropy subject to these constraints, thus, the one maximizing $-\sum_D (\Pr(D) \cdot \log \Pr(D))$. The argument for this choice is that any distribution other than the MaxEnt distribution effectively makes additional assumptions about the data that reduce the entropy. Accordingly, the MaxEnt distribution is the safest bet. Finding the MaxEnt distribution is a convex problem that can be solved efficiently using standard convex optimization techniques [47]. After deriving this distribution, one will be able to compute efficiently $\Pr(\tau \in D)$ for any tile τ, and the information content of τ can then be computed as $\text{IC}(\tau) = -\log(\Pr(\tau \in D))$. Also, the method proposed in [60] seeks for tiles that are easy to assimilate by the user. The assimilation cost of τ is quantified using $\text{DL}(\tau)$ the description length of τ under a Shanon-optimal code. Finally, the subjective interestingness of a pattern is defined as $\text{SI}(\tau) = \frac{\text{IC}(\tau)}{\text{DL}(\tau)}$. Then, one can use some enumeration algorithm in order to identify tiles having the highest score of subjective interestingness.

2.4.2 Incorporation of user preferences

Other methods propose to take into account the user from a different perspective: "the preferences". The simplest case is when the user explicitly expresses her preferences through a set of objective measures $f_1, ..., f_n$ defined on patterns, such as frequency, density, size, etc. When there is only one measure to optimize, the solution can be to extract the top-k patterns w.r.t this measure, given a user specified parameter $k \in \mathbb{N}$. Many methods have been defined to mine top-k patterns regarding the frequency [102], the statistical significance [170], the utility [182], etc. When several objective measures need to be optimized, algorithms have been proposed to identify the so-called skypatterns [57, 196]. A pattern P is a skypattern if it is not dominated by any other pattern regarding all the considered measures to optimize. More formally, P is a skypattern if there is no other pattern P' such that $\forall f_i : f_i(P') \geq f_i(P)$ and $\exists f_i : f_i(P') > f_i(P)$.

In practice, the user may not be able to explicitly define her preferences based on objective measures. However, if a pattern is presented to her, she would definitely know whether this pattern is interesting or not. For this reason, several papers have designed interactive methods that require user feedbacks during the search to learn her preferences. Dzyuba and van Leeuwen [72] use the feedback to guide a diverse beam search in Subgroup Discovery. The beam selection strategy is influenced by the likes and dislikes that the user provides about some previously returned patterns. Bhuiyan and Al Hasan [30] exploit the user's interactions to bias the probability distribution used to sample frequent patterns with Markov Chain Monte Carlo search. Dzyuba et al. [73] propose to frame this problem as an interactive learning and mining loop that consists of four steps shown in Fig. 2.12:

- **Mining patterns:** A set of patterns will be selected using a mining algorithm, and then presented to the user. This algorithm selects these patterns based on some subjective input that represents user-preferences. Generally, this input is initially empty and the mining is objective. Then, in the next iterations, a model of user interests is made progressively more precise and accurate, which allows to discover patterns that are more relevant for this specific user.

- **Interacting with the user:** The patterns are displayed to the user. She can explore them and express her feedback about them. This feedback can be expressed in different manners. For instance, the user can be asked to like and dislike a subset of these patterns [72], some other methods require to rank these patterns from the best to the worst [181, 220]. Expressing the feedback should be simple in order to make the interactive process realistic and not exhausting, but at the same time should allow to derive user's interests.

- **Learning user-specific pattern interestingness:** Use the feedback to build and improve the model of the user interests, i.e., to identify what makes patterns interesting to the user. This model will be used in the next iteration when mining the next set of patterns, to find better patterns regarding the user interests.

- **Repeat:** Mine and interact again relying on the updated score. In other terms, the four steps of the mining process is repeated again using the new interestingness model that includes the new feedbacks.

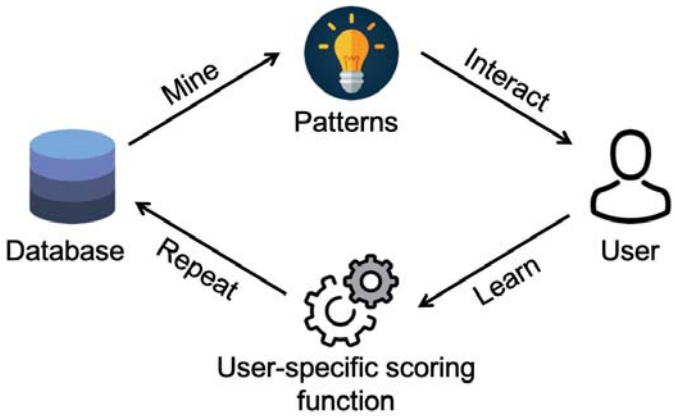


Figure 2.12: Interactive pattern-based data exploration to incorporate user preferences [73].

Similarly, several other methods also require feedbacks from the user to derive her preferences. In [220], the user is asked to rank a small set of patterns according to her interest, and a predictive model (e.g., SVM) is trained based on this feedback and then used to rank the remaining patterns. The authors proposed models designed for itemsets, sequential and structural patterns. For example, to find interesting frequent itemsets, the interestingness of an itemset P is evaluated by a subjective measure R that is based on the difference between the observed frequency $f_o(P)$ and the expected frequency $f_e(P)$. This measure is defined as: $R(f_o(P), f_e(P)) = \log(f_o(P)) - \log(f_e(P))$. While $f_o(P)$ is constant and depends only in the input dataset, $f_e(P)$ is a function having some unkonwn parameters that need to be trained using the elicited feedbacks. From the ranking of patterns, this method considers that $P_i >_f P_j$ if the user judges P_i more interesting. This will be expressed as the following inequality constraint $R(f_o(P_i), f_e(P_i)) > R(f_o(P_j), f_e(P_j))$. Thus, the goal will be to infer the values of parameters of f_e so that the number of violated inequality constraints is minimized. The authors used a ranking SVM formulation to solve this problem. A similar approach is proposed in [181] for the task of Subgroup Discovery, where a model is trained based on the subgroups that the user marked as interesting.

2.5 Discussion

From a general perspective, a common point of vertex-attributed graph mining methods presented in Section 2.2, except outlier detection, is that they aim to find regularities instead of peculiarities. For example, community detection approaches find groups of highly connected vertices having similar attribute values. Most of local pattern mining methods for this kind of graphs aim to find dense subgraphs that have a subspace of attributes with similar values. These works focus their attention on the similarity

inside the subgraphs, while underestimating exceptionality of the subgraph characteristics with respect to the whole graph. Interestingnly, Atzmueller et al. [15] propose to mine descriptions of communities from vertex attributes with a Subgroup Discovery approach. However, the setting of this approach is supervised since each community is treated as a target attribute. In Chapter 3, we propose a method with a similar purpose for mining exceptional attributed subgraphs, but in an unsupervised setting. The notion of target is more dynamic and it corresponds to some subset of vertex-attributes enumerated by the mining algorithm.

There is a myriad of methods that involve the user into the mining process, however, none of them proposes a model for mining local patterns in vertex-attributed graphs. While De Bie [59] defines a general subjective interestingness framework, adapting this framework for the complex structure of vertex-attributed graphs is not straightforward. In Chapter 4, we propose the first approach that exploits this framework to integrate both the graph structure and vertex attributes, in order to mine attributed subgraphs that are subjectively interesting. Furthermore, following the same steps of preference-based interactive mining defined in [73], we propose in Chapter 5 an approach that iteratively integrates user preferences to determine subgraphs of interest in vertex-attributed graphs. This approach is defined for the task of geolocated event detection in social media. In fact, this task can target a large number of people who are able to undestand its result and interact with it. This makes it possible to perform experiments with many real users to evaluate this approach.

In several vertex-attributed graphs, a hierarchical relationship stands between attributes. In practice, attributes of different level of the hierarchy are related and they share overlapping information. In other terms, if an attribute a_i is a parent of another attribute a_j, telling the user that some subgraph is characterized by a_j would let her infer some information about a_i in this subgraph. Integrating the dependency between attributes into the subjective interestingness has not been studied yet in the literature. In Chapter 6, we address this problem and we propose a model that considers the inferences that the user would make when attributes are hierarchical. To simplify, we study this problem independently from any graph structure. However, one of potential future works is to apply it for assessing interestingness of subgraphs where vertices are described by hierarchical attributes.

CHAPTER 3

Exceptional Attributed Subgraph Mining

Contents

3.1 Introduction

As shown in the previous chapter, a large number of methods have been proposed to extract local patterns from vertex-attributed graphs. The majority of these approaches aim to identify dense subgraphs with homogeneous attributes. In order to ensure that the found subgraphs are dense, structural constraints are required on them, for example, subgraphs are required to be quasi-cliques [190, 191] or pseudo-cliques [98, 156]. Some methods do not impose the subgraph to be dense, but only to be connected [86, 187]. The homogeneity of attributes is ensured by similarity measures defined on subspaces of attributes, or by hard constraints on attribute values.

However, most of these works focus their attention on the similarity inside the subgraphs, while underestimating exceptionality of the subgraph characteristics with respect to the whole graph. In fact, a subgraph is likely to be interesting if it is characterized by some properties that distinguish it from the whole graph, otherwise, this subgraph would be already expected from the overall graph properties. To overcome this limitation, we introduce the task of mining exceptional attributed subgraphs. We address the problem of mining connected subgraphs whose vertices share characteristics that are different from those of the remaining vertices. This problem is central in this thesis, and it will be considered again in Chapters 4 and 5, with the purpose of improving patterns quality by incorporating the user's background knowledge (Chapter 4) or preferences (Chapter 5) into the interestingness model. The approach we propose in this chapter is rooted in Subgroup Discovery (SD) [130] and Exceptional Model Mining (EMM) [71]. In both these frameworks, the target concept (e.g., attributes) used to evaluate subgroup's exceptionality is generally fixed throughout all the mining task. In our work, the target is dynamic and it corresponds to some subset of attributes enumerated by the mining algorithm. We aim to identify the exceptional subgraphs with respect to each enumerated subset of target attributes.

Throughout different parts of this chapter, we use an example of application related to mining urban data in order to illustrate the concepts that we will define. More specifically, this example consists in mining data provided by social networks (such as Foursquare and Google Place) to identify geographic areas with homogeneous and exceptional characteristics. The identified areas are described by their associated characteristics that distinguish them from the rest of the city. The vertex-attributed graph model is illustrated in Fig. 3.1. While we apply the proposed approach to only urban data in this chapter, this approach remains generic and it can be used in numerous other applications. For example, Moranges et al. [155] used this method to study relationships between odor perception and brain areas based on fMRI data.

Exceptional attributed subgraph mining is a novel graph mining task that exploits both the contrast of vertices attributes and the graph structure through a connectivity constraint. We propose two algorithms to discover exceptional subgraphs. The first one is an exhaustive algorithm that is based on a constraint-based enumeration strategy. It uses original and efficient upper bounds and some other techniques to reduce the search space. It takes benefit from a closure operator to reduce redundancy and efficiently prune the search space. The second algorithm mines closed exceptional subgraphs by directly sampling the space of closed patterns in a similar way as [37, 106, 183].

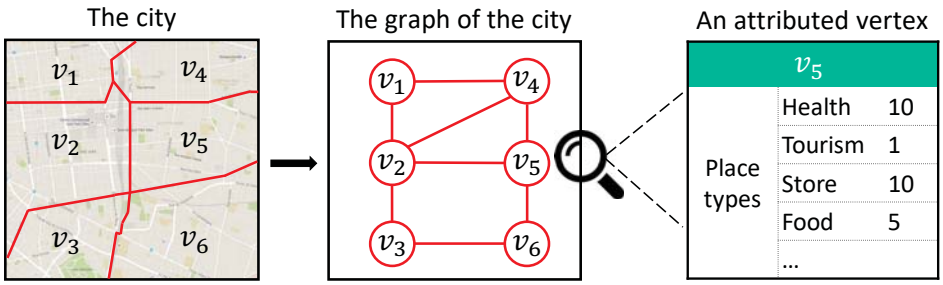


Figure 3.1: Example of a graph modeling a city.

This chapter is organized as follows. Sections 3.2 formally introduces the problem of exceptional attributed subgraph mining. The algorithms proposed to solve this problem are presented in Section 3.3. We report in Section 3.4 a thorough empirical study performed on real-world dataset, including (1) a demonstration of the efficiency of the used pruning techniques, (2) the impact of the parameters and the input graph dimensions on the performance of the algorithms, and (3) the relevance of the discovered results. Our conclusions are drawn in Section 3.5.

3.2 Exceptional attributed subgraph mining problem

In this section, we provide the necessary definitions and terminology. We assume given a vertex-attributed graph $G = (V, E, \hat{A})$. V is the set of vertices, $E \subseteq V \times V$ is the set of edges, and $\hat{A}$ is the set of numerical attributes on vertices (formally, functions mapping a vertex onto an attribute value), with $\hat{a}(v) \in Dom_a$ denoting the value of attribute $\hat{a} \in \hat{A}$ on $v \in V$. We use hats in $\hat{a}$ and $\hat{A}$ to signify the empirical values of the attributes observed on the graph G. The notations a and A represent the set of attributes in a more generic way, they are used to denote (possibly random) variables over the same domains, and to denote the pattern syntax. The values of $\hat{A}$ can be aggregated over a subset of vertices $U \subseteq V$ and a subset of attributes $A' \subseteq A$: $sum(U, A') = \sum_{v \in U} \sum_{a \in A'} \hat{a}(v)$. To simplify the notation, we use $sum(U)$ to denote $sum(U, A)$.

As an example, Fig. 3.1 and Fig. 3.2 present a graph derived from the division of a city into 6 areas (from v_1 to v_6). Nearby venues are grouped into small areas (geographers generally use tiles of 200 meters) over which venue characteristics are aggregated into count data. These areas are hereafter considered as the vertices V of a graph $G = (V, E, \hat{A})$ whose edges E connect adjacent areas (that share a part of their borders), $\hat{a}(v)$ is the number of venues of an attribute $\hat{a}$ in the area associated to a vertex v. The area represented by v_1 is adjacent to the ones represented by v_2 and v_4, and consequently an edge connects v_1 to v_2 and another one v_1 to v_4. The numbers of venues of each attribute in a given area makes the vector associated to the corresponding vertex. The distribution of venue categories $A = \{Health, Tourism, Store, Food\}$ is detailed in Fig. 3.2. $sum(\{v_1\}, health) = 1$ as there is one venue with the category *health* in the area associated to v_1. We can also observe that $sum(\{v_1\}, \{Health, Tourism, Store, Food\}) = 22$, and for the set $U = \{v_2, v_5\}$, $sum(U) = 49$.

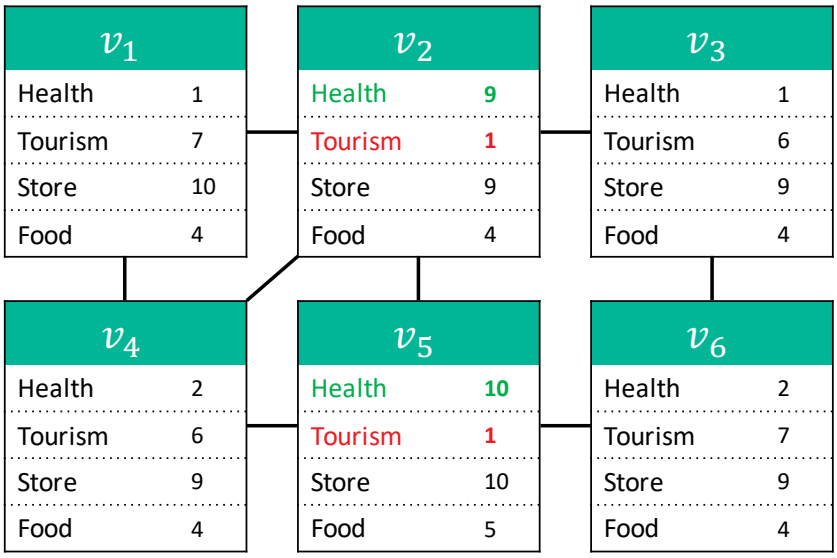


Figure 3.2: Example of the distribution of venues in areas.

Our goal is to discover connected subgraphs associated to exceptional attributes. An attribute is exceptional for a subgraph if it is more frequent in its vertices than in the remaining of the graph. The scarcity of an attribute can also be a relevant element to describe a subgraph. For example, in Fig. 3.2, vertices v_2 and v_5 have a surplus on the attribute *Health* compared to the rest of the graph, while having a loss on the attribute *Tourism*. We formalize the excess and deficit in the amount of some attributes by means of characteristics defined as follows.

Definition 3.1 (Characteristic). A characteristic is a pair $S = (S^+, S^-)$ with S^+ and S^- two disjoint subsets of A. Also, given two characteristics $S_1 = (S_1^+, S_1^-)$ and $S_2 = (S_2^+, S_2^-)$, we have:

- $S_1 \cap S_2 = (S_1^+ \cap S_2^+, S_1^- \cap S_2^-)$,
- $S_1 \cup S_2 = (S_1^+ \cup S_2^+, S_1^- \cup S_2^-)$,
- $S_1 \subseteq S_2 \Leftrightarrow S_1^+ \subseteq S_2^+ \wedge S_1^- \subseteq S_2^-$,
- $|S| = |S^+ \cup S^-| = |S^+| + |S^-|$, (since $S^+ \cap S^- = \emptyset$).

In order to assess the relevancy of the characteristic S with respect to the subgraph induced by $U \subseteq V$, noted $G[U]$, we define the measure $WRAcc(U, S)$, an adaptation of the weighted relative accuracy measure widely used in Subgroup Discovery [130]. $WRAcc(U, S)$ measures to which degree the prevalence of attributes of S^+ (resp. S^-) are higher (resp. lower) than expected in vertices of U.

A set of attributes $A' \subseteq A$ is discriminant to the induced subgraph $G[U]$ if it is more or, on the contrary, less frequent in $G[U]$ than in G. This is evaluated by the $gain$ function:

$$gain(U, A') = \frac{sum(U, A')}{sum(U)} - \frac{sum(V, A')}{sum(V)}.$$

The validity of a characteristic $S = (S^+, S^-)$ with respect to $G[U]$ is assessed by the predicate $valid(U, S)$. This predicates is true if each vertex $v \in U$ has a positive gain for each attribute $a \in S^+$, and a negative gain for each category $a \in S^-$. It is formally defined as:

$$valid(U,S) \equiv \bigwedge_{v \in U} \Big(\big(\bigwedge_{a \in S^+} \delta_{gain(a,v)>0} \big) \bigwedge \big(\bigwedge_{a \in S^-} \delta_{gain(a,v)<0} \big) \Big).$$

The quality of a characteristic S can be globally measured by $gain(S^+, U) - gain(S^-, U)$. However, a major drawback of the gain is that it is easy to obtain high value with highly specific characteristics [130], more precisely characteristics associated to a small set of vertices. Weighted relative accuracy makes a trade-off between generality and gain by considering the relative size of the subgraph.

$$WRAcc(U,S) = \begin{cases} (gain(S^+,U) - gain(S^-,U)) \times \frac{sum(U)}{sum(V)}, & \text{if } valid(U,S) \\ 0, & \text{otherwise.} \end{cases}$$

The main differences with the $WRAcc$ used in Subgroup Discovery [130] are (1) this adapted $WRAcc$ considers both the positive and the negative contrasts in an unsupervised setting (i.e., there is no class attribute in our setting, the "target" is settled by each pattern), (2) it takes into account the homogeneity of the characteristic S on vertices of U, using the predicate $valid(U, S)$.

We now define the pattern domain we consider:

Definition 3.2 (Exceptional subgraph). Given a graph $G = (V, E, \hat{A})$ and two thresholds σ and δ, an exceptional subgraph (U, S) is such that (1) $|U| \geq \sigma$, (2) $G[U]$ is connected, and (3) $WRAcc(U, S) \geq \delta$.

Given an exceptional subgraph (U, S), a large number of included exceptional subgraphs can be derived, i.e. patterns (U', S') such that $U' \subseteq U$ and $S' \subseteq S$. As these patterns (U', S') are already described and covered by (U, S), they unnecessarily increase the size of the solution set. This redundancy can be avoided thanks to a closure operator [127] defined below.

Definition 3.3 (Formal concept). Let max_S and max_V be two operators forming a Galois connection:

- $max_S : 2^V \rightarrow 2^A \times 2^A$, that provides the most specific characteristic associated to the subgraph induced by $U \subseteq V$:

$$max_S(U) = \Big(\{a \in A \mid \bigwedge_{v \in U} \delta_{gain(a,v)>0}\}, \{a \in A \mid \bigwedge_{v \in U} \delta_{gain(a,v)<0}\} \Big).$$

- $max_V : 2^A \times 2^A \rightarrow 2^V$, that returns the set of vertices supporting the characteristic S:

$$max_V(S) = \{v \in V \mid valid(S, \{v\})\}.$$

Notice that $max_S \circ max_V$ and $max_V \circ max_S$ give two closure functions, i.e., they are extensive, monotonic, and idempotent. A pair (U, S), where S is a characteristic and $U \subseteq V$, is a formal concept iff $S = max_S(max_V(S))$ and $U = max_V(S)$, or equivalently, $S = max_S(U)$ and $U = max_V(max_S(U))$.

It may happen that a formal concept as defined above does not correspond to a connected subgraph. For example, in Fig. 3.3, (U, S) is a formal concept, with $U = \{v_1, v_3, v_4, v_6\}$ and $S = (\{c_1\}^+, \{c_2\}^-)$. However, (U, S) is not an exceptional subgraph because $G[U]$ is not connected. Maximal patterns address this limitation:

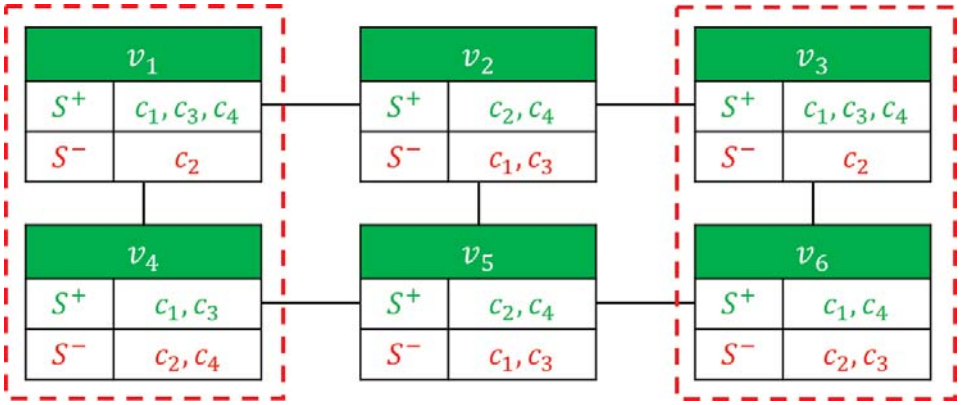


Figure 3.3: Example of a formal concept (U, S), with $S = (\{c_1\}^+, \{c_2\}^-)$, $U = \{v_1, v_3, v_4, v_6\}$ such that $G[U]$ is not connected.

Definition 3.4 (Maximal pattern). A set of maximal patterns is derived from a formal concept (U, S) as:

$$\{(CC, max_S(CC)) \mid CC \text{ is a connected component of } G[U]\}$$

In other terms, a maximal pattern $(CC, max_S(CC))$ is made of the most specific characteristic for CC, but also, the connected subgraph $G[CC]$ cannot be extended to another connected subgraph while keeping the current characteristic $max_S(CC)$.

Following the example from Fig. 3.3, the formal concept (U, S) contains two connected components $CC_1 = \{v_1, v_4\}$ and $CC_2 = \{v_3, v_6\}$, with $max_S(CC_1) = (\{c_1, c_3\}^+, \{c_2\}^-)$ and $max_S(CC_2) = (\{c_1, c_4\}^+, \{c_2\}^-)$. From these two connected components, two maximal patterns $(max_S(CC_1), CC_1)$ and $(max_S(CC_2), CC_2)$ are derived.

Finally, all these definitions are used to establish the notion of closed exceptional subgraph:

Definition 3.5 (Closed exceptional subgraph). Let S be a characteristic and $U \subseteq V$ a subset of vertices, (U, S) is a closed exceptional subgraph iff (1) (U, S) is a maximal pattern (2) (U, S) is an exceptional subgraph.

The rest of the chapter is devoted to the computation and evaluation of the complete set of closed exceptional subgraphs. This requires searching for two combinatorial search spaces, with constraints that cannot be used according to the usual techniques of search space pruning. Thus, a naive approach cannot achieve this task for large graphs or a large number of categories. In the following, we propose an efficient approach that takes benefit from closed pattern properties.

3.3 Computing exceptional subgraphs

This section introduces two distinct approaches to extract closed exceptional subgraphs. First, we present an exact algorithm that aims at discovering the complete set of closed exceptional subgraphs. Second, we devise a heuristic algorithm that samples the space of closed exceptional subgraphs within a user-defined time-budget. This approach makes possible to obtain instant results and to successfully scale up on datasets with a large number of attributes.

3.3.1 Exhaustive search

In order to enumerate the set of all closed exceptional subgraphs, the algorithm CEnergetics (see Algorithm 1) explores the space of characteristics $S = (S^+, S^-)$, and for each characteristic, it enumerates the maximal patterns that can be generated from S using the closure operators. We start from an empty characteristic $(S^+, S^-) = (\emptyset, \emptyset)$ and consider the candidate categories that can be used to expand S: $X = (X^+, X^-)$: X^+ contains the categories that can be added to S^+, and X^- the ones that can be added to S^-. $Y \subseteq V$ represents a set of vertices that verifies $valid(Y, S)$. Initially, Y contains all the vertices V, and $(X^+, X^-) = (C, C)$. In each recursive call of CEnergetics, S is extended with an element x of X^+ or of X^-. Y is then reduced to the vertices v that satisfy $valid(\{v\}, S \cup \{x\})$.

The predicate *valid* is anti-monotone with respect to the inclusion of characteristics: Considering two characteristics S_1, S_2 such that $S_1 \subseteq S_2$ and $U \subseteq V$, we have $valid(U, S_2) \Rightarrow valid(U, S_1)$. By the contraposition, the invalid vertices for S_1 are also invalid for S_2, and therefore, the valid set of vertices associated to $S \cup \{x\}$ is a subset of Y (a property used in Line 5 of Algorithm 1). We also take benefit from this anti-monotony using the fail first principle: To extend the current characteristic S, we choose the characteristic x for which the set $\{v \in Y \mid valid(\{v\}, S \cup \{x\})\}$ is the smallest. After updating Y, we explore each connected component CC of $G[Y]$ independently and form $(CC, max_S(CC))$ that is, by definition, a maximal pattern. If $max_S(CC) \subseteq S \cup X$, then the maximal pattern $(CC, max_S(CC))$ has not yet been explored and CEnergetics is recursively called with $S = max_S(CC)$ and $Y = CC$ (Line 9). This allows to explore only characteristics S and vertices subsets Y that form maximal patterns (Y, S), and without redundancy.

Another pruning mechanism is used on Line 2 where the function UB is used to upper bound the $WRAcc$ measure. This function relies on the aggregation property of the $WRAcc$ measure as defined below.

Property 3.1. Let $S = (S^+, S^-)$ be a characteristic, and $U \subseteq V$ a set of vertices satisfying $valid(U, S)$, we have:

$$WRAcc(U, S) = \sum_{v \in U} \left(\sum_{x \in S^+} WRAcc\left(v, (x^+, \emptyset)\right) + \sum_{x \in S^-} WRAcc\left(v, (\emptyset, x^-)\right) \right).$$

Algorithm 1: CENERGETICS(S, X, Y, R, δ, σ)

Input: $S = (S^+, S^-)$ the current explored characteristic, $X = (X^+, X^-)$ the candidate sets, Y a connected component s.t (Y, S) is a maximal pattern
Output: R the result set under construction

```
if X ≠ (∅, ∅) then
    if |Y| ≥ σ and UB(Y, S ∪ X) ≥ δ then
        // Extending S using the fail first principle:
        x ← argmin_{x∈X} |{v ∈ Y | valid({v}, S ∪ {x})}|
        Y' ← {v ∈ Y | valid({v}, S ∪ {x})}
        for each connected component CC ⊆ G[Y'] do
            if max_S(CC) ⊆ S ∪ X then
                // (CC, max_S(CC)) has not been explored yet
                CENERGETICS(max_S(CC), X \ max_S(CC), CC, R, δ, σ)
        CENERGETICS(S, X \ {x}, Y, R, δ, σ)
else
    if |Y| ≥ σ and WRAcc(Y, S) ≥ δ then
        R ← R ∪ {(Y, S)}
```

Proof. Since $valid(U, S)$ is $true$:

$$
\begin{aligned}
&WRAcc(U,S) = (gain(S^+,U) - gain(S^-,U)) \times \frac{sum(U)}{sum(V)},\\
&= \left(\frac{sum(U,S^+)}{sum(U)} - \frac{sum(V,S^+)}{sum(V)} - \frac{sum(U,S^-)}{sum(U)} + \frac{sum(V,S^-)}{sum(V)}\right) \times \frac{sum(U)}{sum(V)},\\
&= \left(\frac{sum(U,S^+)}{sum(V)} - \frac{sum(V,S^+)\cdot sum(U)}{sum(V)^2}\right) + \left(\frac{sum(V,S^-)\cdot sum(U)}{sum(V)^2} - \frac{sum(U,S^-)}{sum(V)}\right),\\
&= \sum_{v\in U}\left(\left(\frac{sum(v,S^+)}{sum(V)} - \frac{sum(V,S^+)\cdot sum(v)}{sum(V)^2}\right) + \left(\frac{sum(V,S^-)\cdot sum(v)}{sum(V)^2} - \frac{sum(v,S^-)}{sum(V)}\right)\right),\\
&= \sum_{v\in U}\left(\sum_{x\in S^+}\left(\frac{sum(v,x^+)}{sum(V)} - \frac{sum(V,x^+)\cdot sum(v)}{sum(V)^2}\right) + \sum_{x\in S^-}\left(\frac{sum(V,x^-)\cdot sum(v)}{sum(V)^2} - \frac{sum(v,x^-)}{sum(V)}\right)\right),\\
&= \sum_{v\in U}\left(\sum_{x\in S^+} WRAcc\left(v,(x^+,\emptyset)\right) + \sum_{x\in S^-} WRAcc\left(v,(\emptyset,x^-)\right)\right).
\end{aligned}
$$

From this property, we can derive the following function UB and demonstrate that it can be used to upper bounds the $WRAcc$ value.

Definition 3.6 (UB). Let $S = (S^+, S^-)$ be a characteristic, and $U \subseteq V$. $UB(U, S)$ is defined as:

$$UB(U,S) = \sum_{v\in U}\left(\sum_{x\in S^+} WRAcc\left(v,(x^+,\emptyset)\right) + \sum_{x\in S^-} WRAcc\left(v,(\emptyset,x^-)\right)\right).$$

Property 3.2. For each pattern (U_2, S_2) such that $U_2 \subseteq U$ and $S_2 \subseteq S$, we have

$$UB(U,S) \geq WRAcc(U_2,S_2).$$

Proof. (1) If $valid(U_2, S_2) = false$, the property is verified because $UB(U, S) \geq 0$. In fact, UB is a sum of $WRAcc$ values that are always positive or null.

(2) If $valid(U_2, S_2) = true$:

$$
\begin{aligned}
UB(U,S) = & \sum_{v \in U_2} \left(\sum_{x \in S_2^+} WRAcc\left(v, (x^+, \emptyset)\right) + \sum_{x \in S_2^-} WRAcc\left(v, (\emptyset, x^-)\right) \right), \\
& + \sum_{v \in U \setminus U_2} \left(\sum_{x \in S_2^+} WRAcc\left(v, (x^+, \emptyset)\right) + \sum_{x \in S_2^-} WRAcc\left(v, (\emptyset, x^-)\right) \right), \\
& + \sum_{v \in U} \left(\sum_{x \in \in S^+ \setminus S_2^+} WRAcc\left(v, (x^+, \emptyset)\right) + \sum_{x \in \in S^- \setminus S_2^-} WRAcc\left(v, (\emptyset, x^-)\right) \right), \\
& = WRAcc(U_2, S_2) + \sum_{v \in U \setminus U_2} \left(\sum_{x \in S_2^+} WRAcc\left(v, (x^+, \emptyset)\right) + \sum_{x \in S_2^-} WRAcc\left(v, (\emptyset, x^-)\right) \right), \\
& + \sum_{v \in U} \left(\sum_{x \in \in S^+ \setminus S_2^+} WRAcc\left(v, (x^+, \emptyset)\right) + \sum_{x \in \in S^- \setminus S_2^-} WRAcc\left(v, (\emptyset, x^-)\right) \right), \\
& \geq WRAcc(U_2, S_2).
\end{aligned}
$$

Since all the enumerated patterns $P = (U_2, S_2)$ by CENERGETICS satisfy $U_2 \subseteq S \cup X$ and $U_2 \subseteq Y$, we have always $UB(Y, S \cup X) \geq WRAcc(U_2, S_2)$. Thus, if $UB(Y, S \cup X) < \delta$, we discard the current search space.

The number of closed exceptional subgraphs can be exponential in the size of the input dataset. As CENERGETICS enumerates the complete set of closed exceptional subgraph, the number of enumeration steps can be exponential too. However, each recursive call has a worst case time complexity in $O(\max\{|A| \times |V|, |V| + |E|\})$:

- Computing UB on Line 2 or $WRAcc$ on Line 12 take $O(|S \cup X| \times |Y|)$ i.e. $O(|A| \times |V|)$ in the worst case
- The computation of the next candidate $x \in X$ with the fail first principle (Line 4) requires in the worst case $O(|A| \times |V|)$
- Line 5 takes $O(|Y|)$, that is to say $O(|V|)$ at most
- Line 6, computing the connected components, takes $O(|V| + |E|)$
- Line 7, $max_S(CC)$ is obtained in $O(|A| \times |CC|)$. For all the connected components of $G[Y']$, this requires in overall $O(|A| \times |Y'|)$, which corresponds to $O(|A| \times |V|)$ in the worst case.

CENERGETICS enumerates maximal patterns in a depth-first manner. The search space can be represented as a tree where each enumerated maximal pattern (Y, S) corresponds to a single leaf. The depth of this tree is bounded by $2 \times |A|$, since in each recursive call at least one element $x \in X$ is added to S. Thus, the number of recursive calls between two leaves is bounded by $4 \times |A|$ (we backtrack at most $2 \times |A|$ times and then we go in depth at most $2 \times |A|$ times). Thus, we can conclude that the time delay between the enumeration of two leaves of this tree (two different maximal patterns) is polynomial in $O(|A| \times \max\{|A| \times |V|, |V| + |E|\})$.

3.3.2 Exceptional subgraph space sampling

In practice, end-users want to obtain high-quality patterns in a short amount of time, especially in interactive data mining processes. However, we show in experiments that the runtime of CENERGETICS increases when the graph size or the number of attributes increase, and it may require a considerable time to mine very large graphs. To overcome this issue, we propose an approach that computes a sampling of the closed exceptional subgraphs within a user-given time-budget.

We adapt the randomized pattern mining technique of [37] to exceptional subgraphs discovery. This so-called *Controlled Direct Pattern Sampling* enables the user to specify a time budget and computes a set of high-quality patterns whose size directly depends on the specified amount of time.

The idea consists of sampling the patterns based on a probability distribution that depends on the quality patterns. In a first attempt, we proposed to first sample the characteristics and then derive the associated subgraphs. But this strategy failed in computing patterns with high WRAcc values because the graph structure was neglected. Thus, we adopted the reverse approach that consists in randomly generating maximal patterns (U, S).

We perform a random walk on a graph whose vertices are the maximal patterns and the edges connect couple of patterns (U_1, S_1) and (U_2, S_2) such that $U_1 \subseteq U_2$ and there does not exist a maximal pattern (U, S) such that $U_1 \subset K \subset U_2$ (strict inclusion).

To define how is constructed the graph on which the random walk is performed, we need to introduce a new closure function $clo : 2^V \rightarrow 2^V$: Given a connected subgraph induced by U, $clo(U)$ returns the part of $max_V(max_S(U))$ that is connected and contains U:

$$clo(U) = \{v \in max_V(max_S(U)) \mid \exists U' \subseteq max_V(max_S(U)) : U \cup \{v\} \subseteq U' \wedge G[U'] \text{ is connected}\}$$

$clo(U)$ can be computed by extending U recursively with all neighbors v that maintain $max_S(U \cup \{v\}) = max_S(U)$. During the random walk, edges (transitions) are chosen following a probability measure that favors high-quality patterns:

1. The random walk starts by drawing a first vertex using the probability distribution defined as $\mathcal{P}(\{v\}) = \frac{WRAcc(clo(\{v\}), max_S(\{v\}))}{\sum_{u \in V} WRAcc(clo(\{u\}), max_S(\{u\}))}$ in order to form the first explored maximal pattern $(clo(\{v\}), max_S(\{v\}))$.

2. A new maximal pattern is generated from the pattern (U, S) by considering all maximal patterns that are direct super-sets of U. Such patterns are generated by alternatively adding a neighbor element $v \in N(U) \setminus U$ to U and considering the closure $clo(K \cup \{v\})$. $N(U)$ is the set of neighbors of U: $N(U) = \{v \in V \mid \exists u \in U \ : \ (u, v) \in E\}$. (U, S) is also considered among the patterns that can be generated in the next step. The set $Next(U)$ of all possible next subgraphs is then:

$$Next(U) = \{U\} \cup \{clo(U \cup \{v\}) \mid v \in N(U) \setminus U\}.$$

Thus, from $Next(U)$, all the direct successors to (U, S) can be enumerated by:

$$\{(U', S') \mid U' \in Next(U) \text{ and } S' = max_S(U')\}.$$

The next random step is drawn based on the probability $\mathcal{P}(U' \mid U)$, that is the probability to reach $U' \in Next(U)$ from U:

$$\mathcal{P}(U' \mid U) = \frac{WRAcc(U', max_S(U'))}{\sum_{U_2 \in Next(U)} WRAcc(U_2, max_S(U_2))}.$$

This distribution of probabilities rewards transitions toward maximal patterns with large values of $WRAcc(U', max_S(U'))$.

3. The current random walk stops when $U' = U$ and a new one is started from step (1). Otherwise, the random walk continues by repeating Step (2) on the set of vertices U'. At each step of the random walk, if $WRAcc(U, max_S(U)) \geq \delta$ and $|U| \geq \sigma$, the pattern is added to the output result set.

The algorithm EXCESS[1] (see Algorithm 2) samples patterns until the specified execution time is consumed. Since U is extended at each iteration by at least one vertex v, and U is bounded by V, the extension loop (Line 8) stops after at most $|V|$ iterations.

In the following, we prove that all maximal patterns with nonzero $WRAcc$ value have a non zero probability to be generated. To this end, we first prove the following necessary property.

Property 3.3. For each maximal pattern $P = (U, S)$ with $|U| \geq 1$, there exists a maximal pattern $P^\star = (U^\star, S^\star)$ s.t: $U^\star \subset U$ (a strict inclusion) and $\exists v^\star \in N(U^\star) \setminus U^\star$ with $U = clo(U^\star \cup \{v^\star\})$.

Proof. Since we know that for each maximal pattern $P = (U, S)$ with $|U| \geq 1$ there exists a maximal pattern $P' = (U', S')$ s.t $U' \subset U$ (at least the empty pattern $P' = (\emptyset, (C^+, C^-))$), we prove the property by induction: $\forall n \in \mathbb{N}^\star$, for each maximal pattern $P = (U, S)$ such that there exists a maximal pattern $P' = (U', S')$ with $U' \subset U$ and $|U| - |U'| \leq n$, there also exists a maximal pattern $P^\star = (U^\star, S^\star)$ with $U^\star \subset U$ and $\exists v^\star \in N(U^\star) \setminus U^\star$ with $U = clo(U^\star \cup \{v^\star\})$.

- If $n = 1$: $U \setminus U' = \{v\}$, then $clo(U' \cup \{v\}) = clo(U) = U$. Thus $P^\star = P'$,

[1] EXCESS stands for EXceptionnal ClosEd Subgraph Sampler.

Algorithm 2: EXCESS(time_Budget, δ, σ)

Input: time_Budget
Output: R a set of sampled patterns

for $v \in V$ **do**
  if $WRAcc(clo(\{v\}), max_S(\{v\})) \geq \delta$ *and* $|clo(\{v\})| \geq \sigma$ **then**
    $R \leftarrow R \cup (clo(\{v\}), max_S(\{v\}))$
while *current_time* $<$ *time_Budget* **do**
  // Step 1: draw a vertex v
  draw $v \sim \frac{WRAcc(clo(\{v\}), max_S(\{v\}))}{\sum_{u \in V} WRAcc(clo(\{u\}), max_S(\{u\}))}$
  // Step 2: expansion of U
  $U' \leftarrow clo(\{v\})$
  repeat
    $U \leftarrow U'$
    // Compute the set $Next(U)$
    $Next(U) \leftarrow \{U\}$
    for $v \in N(U) \setminus U$ **do**
      $Next(U) \leftarrow Next(U) \cup \{clo(U \cup \{v\})\}$
    for $U' \in Next(U)$ **do**
      if $WRAcc(U', max_S(U')) \geq \delta$ *and* $|U'| \geq \sigma$ **then**
        $R \leftarrow R \cup (U', max_S(U'))$
    draw $U' \sim \frac{WRAcc(U', max_S(U'))}{\sum_{U_2 \in Next(U)} WRAcc(U_2, max_S(U_2))}$
  until $U' = U$;

- Let us suppose that the proposition is true for n. Let $P = (U, S)$ be a maximal pattern for which there exists a maximal pattern $P' = (U', S')$ s.t $U' \subset U$ and $|U| - |U'| \leq n + 1$. If $|U| - |U'| \leq n$, then the proposition is verified according the the induction hypothesis. Otherwise $|U| - |U'| = n + 1$, let $v \in U \cap N(U') \setminus U'$, since $U' \cup \{v\} \subset U$ then $clo(U' \cup \{v\}) \subseteq U$:
  - If $clo(U' \cup \{v\}) = U$, then $P^\star = P'$,
  - If $clo(U' \cup \{v\}) \neq U$. We have $clo(U' \cup \{v\}) \subset U$, and $(max_S(U' \cup \{v\}), clo(U' \cup \{v\}))$ is a maximal pattern, and $|U| - |clo(U' \cup \{v\})| \leq n$. Then, according to the induction hypothesis, the proposition is verified.

∎

Property 3.4. For each maximal pattern $P = (U, S)$ with $WRAcc(U, S) > 0$, the probability $\widetilde{\mathcal{P}}(P)$ that the random walk reaches the pattern P is not null: $\widetilde{\mathcal{P}}(P) > 0$.

Proof. Let us prove by induction on $n \in \mathbb{N}^\star$, that for all maximal pattern $P = (U, S)$ s.t $WRAcc(U, S) > 0$ with $|U| \leq n$: $\widetilde{\mathcal{P}}(P) > 0$.

- For $n = 1$: $U = \{v\}$, and $U = clo(\{v\})$, P can be sampled directly in Step 1:

$$\widetilde{\mathcal{P}}(P) \geq \frac{WRAcc(U, S)}{\sum_{u \in V} WRAcc(clo(\{u\}), max_S(\{u\}))} > 0,$$

- Let us suppose that the proposition is true for n. Let $P = (U, S)$ be a maximal pattern s.t $WRAcc(U, S) > 0$ and $|U| = n + 1$. According to Property 3.3, there exists a maximal pattern $P^\star = (U^\star, S^\star)$ s.t: $U^\star \subset U$ and $\exists v^\star \in N(U^\star) \setminus U^\star$ with $U = clo(U^\star \cup \{v^\star\})$. If $U^\star = \emptyset$, then $U = clo(\{v^\star\})$, this means that P can be sampled on Step 1:

$$\widetilde{\mathcal{P}}(P) \geq \frac{WRAcc(U, S)}{\sum_{u \in V} WRAcc(clo(\{u\}), max_S(\{u\}))} > 0,$$

If $U^\star \neq \emptyset$, since $WRAcc(U, S) > 0$, then $S \neq (\emptyset, \emptyset)$, and we know that $S \subseteq S^\star$, thus $S^\star \neq (\emptyset, \emptyset)$. This means that $WRAcc(U^\star, S^\star) > 0$. In the other hand, $U^\star \leq n$, then $\widetilde{\mathcal{P}}(P^\star) > 0$. Also, $U \in Next(U^\star)$. So, P can be reached after sampling $P^\star$:

$$\widetilde{\mathcal{P}}(P) \geq \widetilde{\mathcal{P}}(P^\star) \times \frac{WRAcc(U, max_S(U))}{\sum_{U_2 \in Next(U^\star)} WRAcc(U_2, max_S(U_2))} > 0.$$

Since each maximal pattern $P = (U, S)$ with $WRAcc(U, S) > 0$ can be reached by the random walk, we can conclude that if $WRAcc(U, S) \geq \delta$ and $|U| \geq \sigma$, then the pattern P has a non zero probability to be returned by EXCESS. Furthermore, the used probability distribution rewards high-quality patterns by giving them more chance to be sampled.

3.4 Experiments

In this section, we report on experimental results to illustrate the interest of the proposed approach. We start by describing the different real-world datasets we use, as well as the questions we aim to answer. Then, we provide a performance study and give some qualitative results. The implementation of the method is in Java and the experiments run on machines equipped with i7-2600 CPUs @ 3.40GHz, and 16GB main memory, running Ubuntu 12.04, and Java Version 1.6. The code and the data are available[2].

3.4.1 Datasets and aims

We considered 10 real-world datasets whose characteristics are given in Table 6.1. Eight of them come from [77] and depict Foursquare venues over 4 US and 4 EU important cities. The venues are described by a hierarchy[3]. We consider the first level (10 attributes) in the first series of experiments and the second level (around 300 attributes) for the second ones. *SF. Crimes* data[4] are provided by a Kaggle challenge and describe the criminal activity in San Francisco. Finally, *San Francisco C&V* is the combination – after normalization – of *SF. Crimes* and Foursquare data over San Francisco. Each city is divided into rectangular zones in such a way that each rectangle contains a minimal number of venues.

In this experimental study, we aim to examine the behaviors of CENERGETICS and EXCESS regarding the following questions:

[2] https://github.com/AnesBendimerad/ClosedExceptionalSubgraphMining
[3] https://developer.foursquare.com/categorytree
[4] https://www.kaggle.com/c/sf-crime

Dataset	$\|V\|$	$\|E\|$	$\|A\|$	#Objects
New York	292	647	10 (356)	71954 venues
Los Angeles	159	348	10 (325)	34504 venues
San Francisco	124	256	10 (328)	21654 venues
Washington	106	216	10 (316)	19190 venues
London	118	241	10 (318)	25029 venues
Paris	115	231	10 (305)	27443 venues
Rome	90	177	10 (279)	13166 venues
Barcelona	109	218	10 (304)	19668 venues
S.F. Crimes	898	2172	39	878049 crimes
S.F. C&V	342	767	49 (328)	878049 cr. + 21654 ven.

Table 3.1: Description of the real-world datasets.

- What is the efficiency of CEnergetics with regard to the graph characteristics that may affect its execution time?
- How effective are CEnergetics' pruning properties?
- Does CEnergetics scale?
- Does EXCESS provide a good sample of Exceptional subgraphs?
- What about the relevancy of Exceptional subgraphs?

No related work, among those presented in Chapter 2, can be used as a competitor of CEnergetics. Indeed, algorithms of pattern extraction in vertex attributed graphs [46, 98, 156, 158, 175, 191] compute dense subgraphs whose vertices have homogeneous attribute values, while CEnergetics focuses on subgraphs whose vertex attributes are different from those of the rest of the graph. Other related works, that look for exceptional subgraphs [121, 135], are designed for graphs with attributes on the edges. Thus, in this section, we compare our two novel algorithms only to the ones of our first attempt [23]: We demonstrate that CEnergetics is more efficient than Energetics (a complete algorithm that extracts non closed exceptional subgraphs) and is able to tackle graphs with more than 150 attributes while Energetics fails with 50 attributes. Furthermore, our new pattern sampler algorithm EXCESS provides better results than EXPreSS. Finally, we report some examples of exceptional subgraphs on real-world data and discuss the insights they convey.

3.4.2 Quantitative study

We compare the efficiency and the effectiveness CEnergetics and Energetics according to the number of attributes and the number of vertices. To this end, we consider the New York graph described in Table 6.1. We vary the number of vertices and attributes by removing or duplicating vertices and attributes. Fig. 3.4 reports the runtime, the number of explored patterns and the number of returned patterns of

both CEnergetics and Energetics on this testbed. The values of parameters are: $\delta = 0.01$, $\sigma = 1$. CEnergetics clearly has an advantage over Energetics. It is much faster, explores a lower number of candidates, and return a much more concise set of patterns. The differences between the two algorithms are more important when the number of attributes varies. CEnergetics outperforms Energetics with several order of magnitudes. Furthermore, CEnergetics is able to handle graphs with more than 150 attributes while Energetics fails as soon as graphs involve more than 40 attributes.

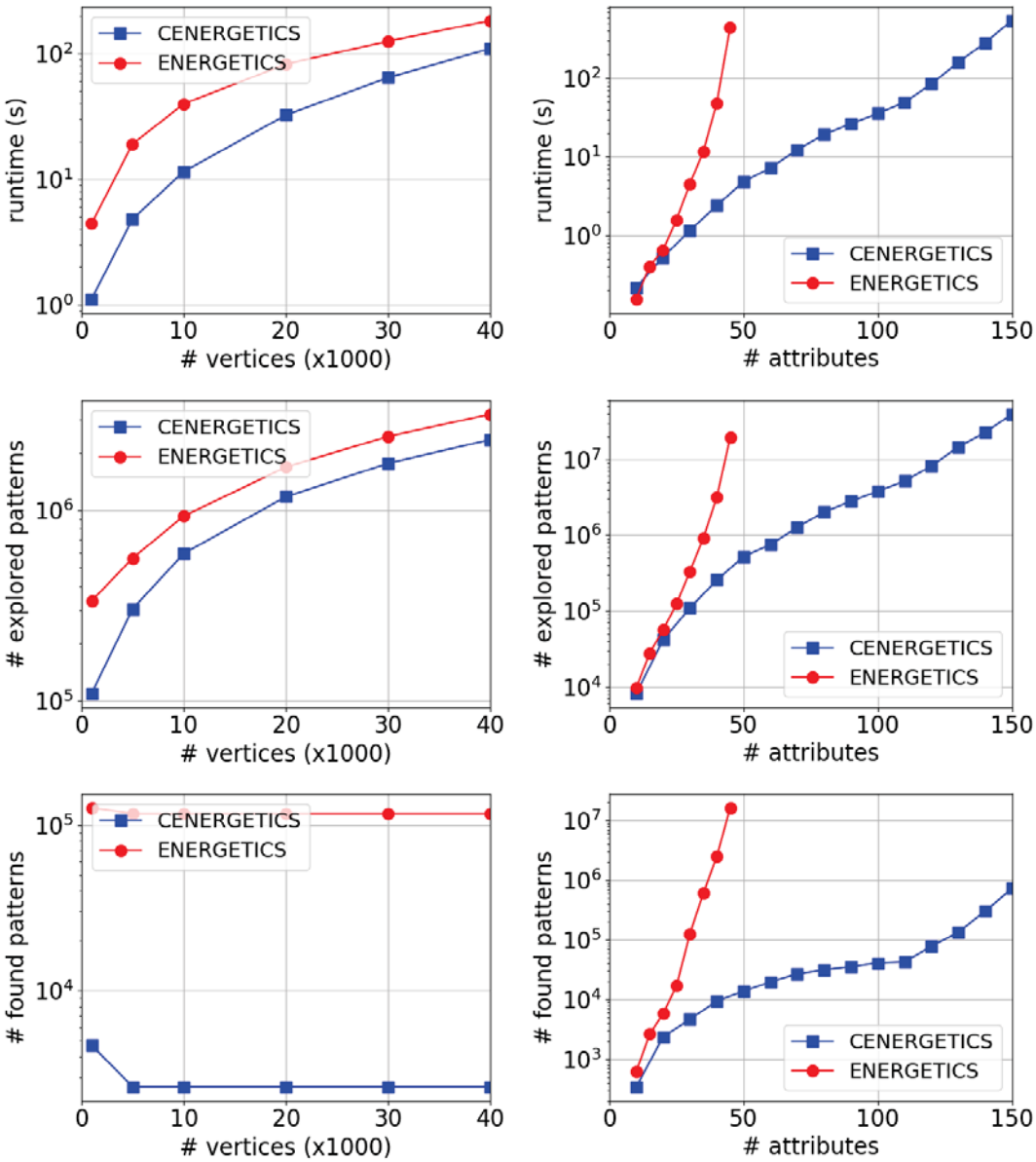


Figure 3.4: Comparison (i.e., runtime, number of explored patterns and number of returned patterns) between CEnergetics and Energetics according to the number of vertices and attributes (default values: number of vertices = 1000, number of attributes = 30).

We now focus on the study of CEnergetics with respect to the parameters of the algorithm (i.e., σ, the minimum number of vertices involved in a pattern, and δ the minimum WRAcc threshold). By default, these values are set to $\delta = 0.01$ and $\sigma = 1$ in order to not being stringent. Fig. 3.5 reports the behavior (i.e., runtime, number of explored sub-graphs and number of patterns) of CEnergetics on the 10 real-world datasets when varying the input parameters δ and σ. The obtained results confirm the previous findings. The execution time and the numbers of explored and returned patterns increase when the thresholds become less stringent. Interestingly, *S.F. C&V* is the dataset whose execution times are the most important. This confirms the previous finding that the number of attributes is the most influential data parameter in the discovery of exceptional subgraphs.

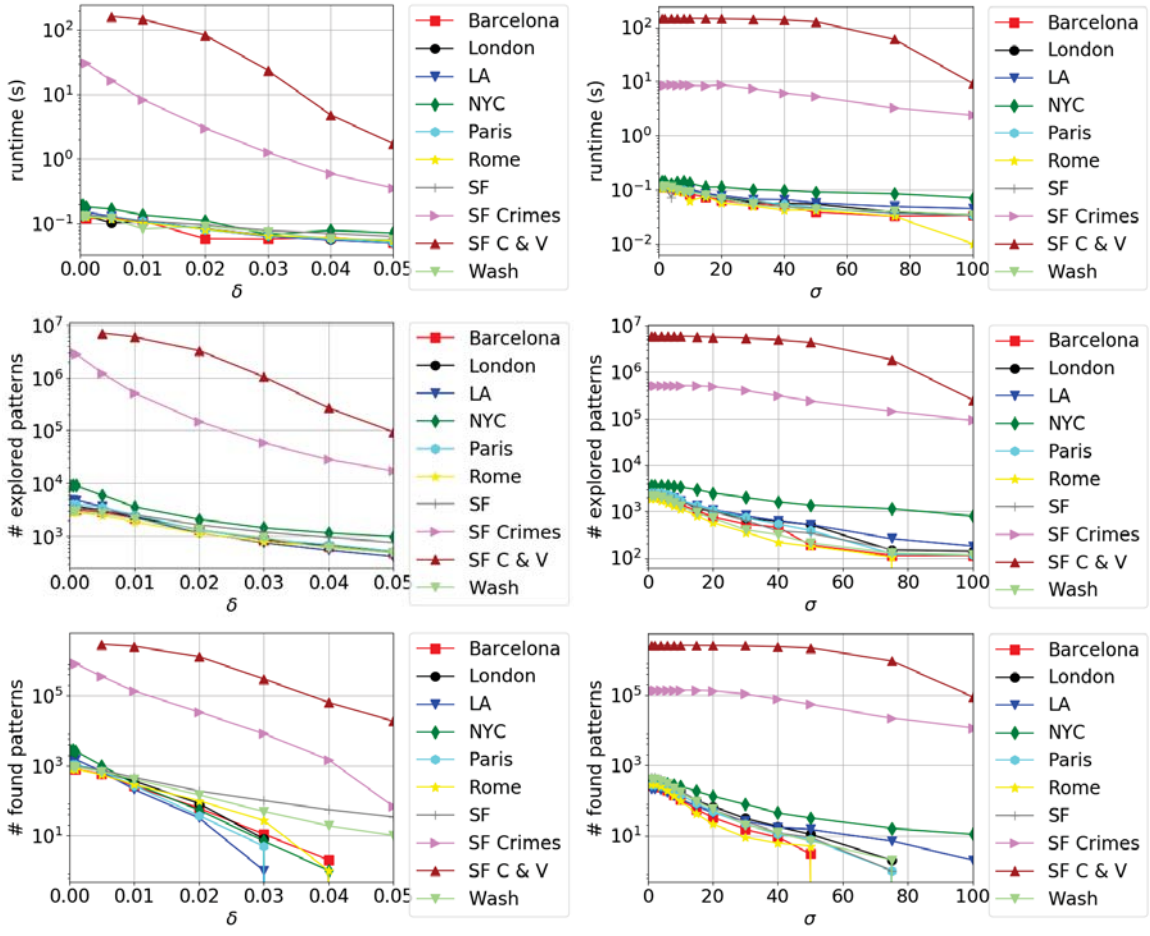


Figure 3.5: Impact of parameters δ (1st column) and σ (2nd column) on CEnergetics (runtime in 1st raw, #explored patterns in 2nd raw, #patterns in 3rd raw). Default values are $\delta = 0.01, \sigma = 1$.

We also study the behaviour of our algorithm with regard to the replication factor. For a replication factor equal to n, the attributed graphs are duplicated n times such that the initial vertices are repeated n times with the same attributes values and the same connections with the corresponding duplicated vertices. Therefore, a n-duplicated attributed graphs correspond to n identical attributed graphs that are

not connected together and thus contains n times the number of exceptional subgraphs of the original graph. For each replicate attributed graph, we compute the ratio of the execution time of CEnergetics on the duplicated graph to the execution time of CEnergetics on the original graph. Figure 3.6 reports this ratio for the 10 replicated graphs. For most of the datasets, the algorithm behaves almost linearly with respect to the replication factor. However, this is not the case for *S.F. C&V* and *S.F. Crimes* that are the datasets with the highest number of attributes. For these two datasets, the performance degrades when the replication factor increases. The runtime ratio increases superlinearly with the replication factor.

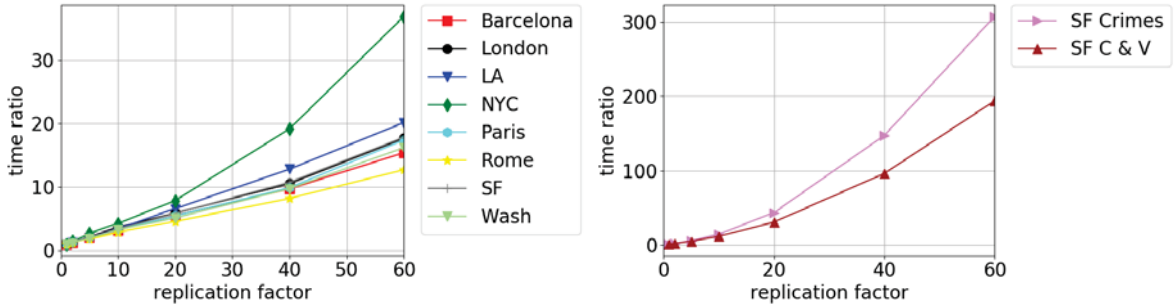


Figure 3.6: Runtime ratio with respect to the replication factor for real world datasets ($\delta = 0.01$ except SF Crimes and SF C&V (0.03 and 0.05), $\sigma = 1$).

In order to demonstrate the effectiveness of the pruning techniques used (the upper bound UB, and the Fail First Principle FFP), we compare the performance of CEnergetics in four different configurations:

1. no opt: in this configuration, none of the pruning techniques is used.
2. FFP: we only use the Fail First Principle (FFP).
3. UB: we only use the upper bound UB.
4. UB+FFP: we use both UB and FFP.

We performed these four configurations on an attributed graph involving 10000 vertices and 30 attributes built by duplicating the NYC Foursquare graph. It is important to note that *no opt* configuration considers closed exceptional subgraphs which makes the extraction feasible. We study the runtime and the number of explored sub-graphs when varying the value of δ. Results are given in Fig. 3.7. UB+FFP outperforms all the other configurations with at least one order of magnitude, especially when the value of δ is increased. Indeed, the use of UB takes benefit from the minimum threshold δ in order to reduce the runtime and the number of explored patterns. These results confirm that even if UB is the most effective technique, the simultaneous consideration of UB and FFP makes the algorithm much more efficient.

These first experiments demonstrate that CEnergetics is only efficient for graphs whose number of attributes is rather small (at most 150). Indeed, CEnergetics is not able to manage attributed graphs

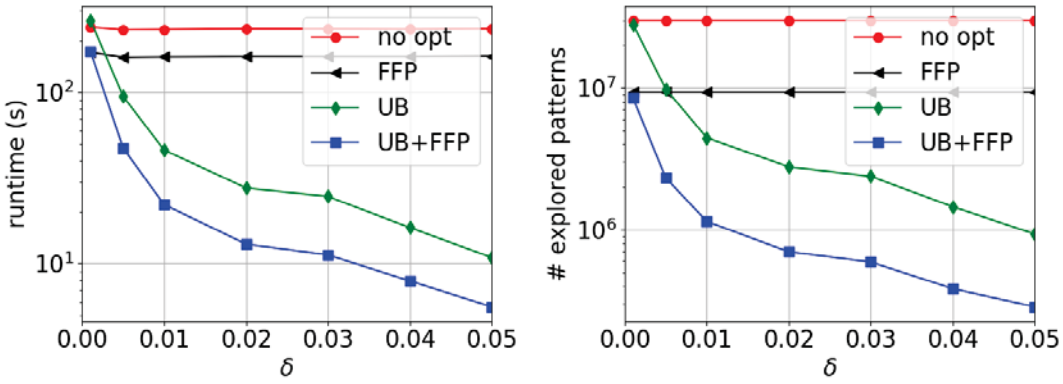


Figure 3.7: Impact of pruning techniques on runtime (1st column), and the number of explored patterns (2nd column). The number of discovered patterns decreases from 10^6 to 140 (not reported on the figures).

with large number of attributes (e.g., hundreds). EXCESS has been designed especially to perform on graphs with hundreds of attributes, using a time budget to control the execution time and the number of computed patterns.

To evaluate the ability of EXCESS to compute exceptional subgraphs of high $WRAcc$ values, we report in Fig. 3.8 the distributions of the WRAcc measure of both the complete set of exceptional subgraphs returned by CEnergetics and the sample provided by EXCESS. Several time budgets are used and they are all lower than the execution time required by CEnergetics. We can observe that the two distributions are similar and the sampling approach succeeds in fostering patterns with high WRAcc measure. Also, the higher the time budget, the better the distribution. Fig. 3.9 reports similar distributions for the real-world datasets with hundreds of attributes for which an exhaustive search is not possible. The distributions are similar. Thus, EXCESS makes it possible to discover high quality patterns within a time-budget.

We also compare EXCESS with EXPreSS [23] which does not take into account closed patterns. Distributions of patterns sampled by each of these approaches are reported in Figure 3.10 using a logarithmic scale. These results reveal that EXCESS returns a larger sampling than EXPreSS for the same time budget. Interestingly, EXCESS provides much more patterns with higher $WRAcc$ values than patterns sampled by EXPreSS. This confirms that EXCESS is able to extract more patterns of better quality (i.e., with high $WRAcc$ values) than EXPreSS.

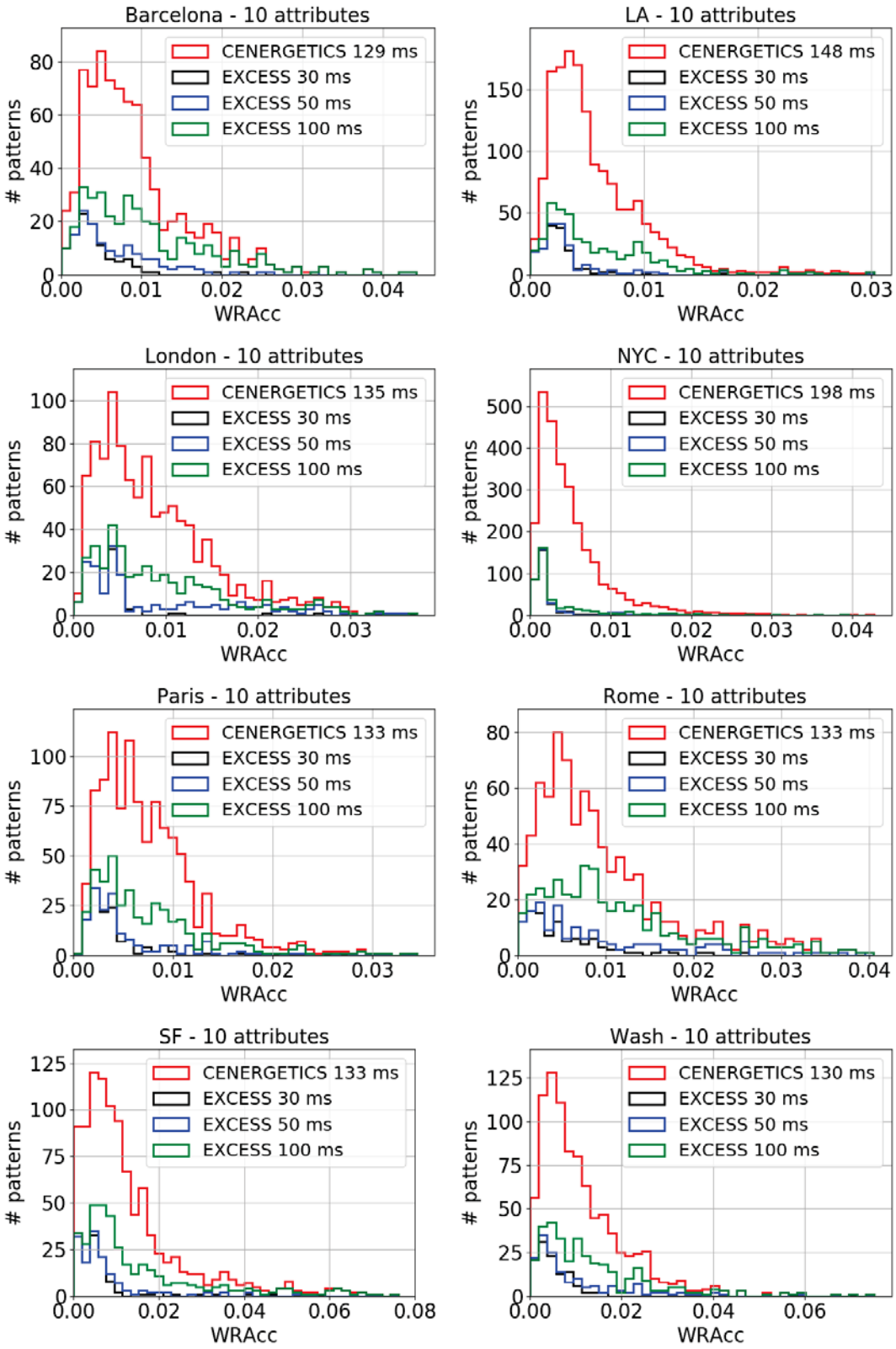


Figure 3.8: Distributions of the patterns from CENERGETICS and EXCESS with different time budgets ($\delta = 0$). The number of attributes is 10.

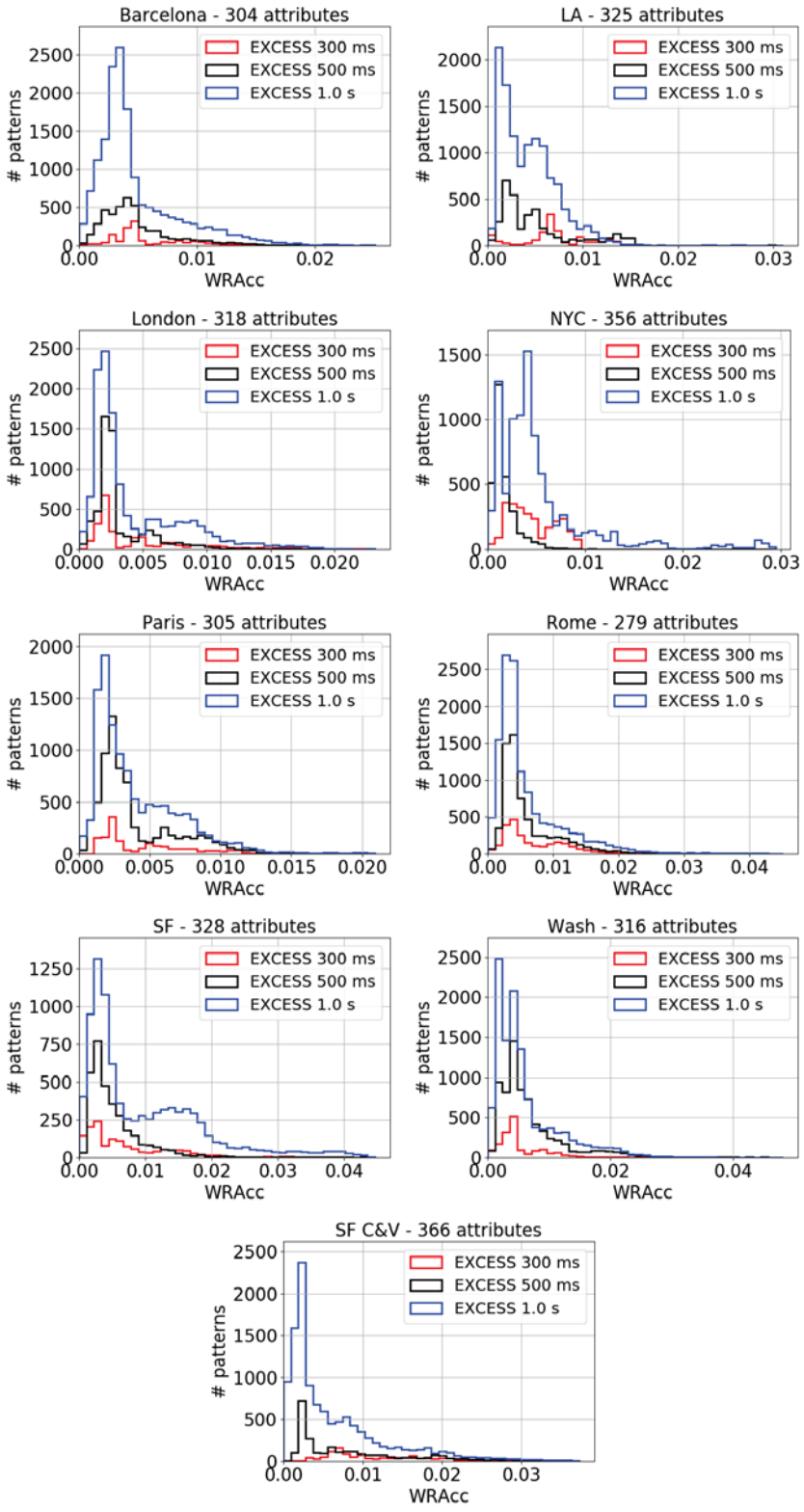


Figure 3.9: Distribution of the output space sampling with different time budgets for datasets with larger number of attributes.

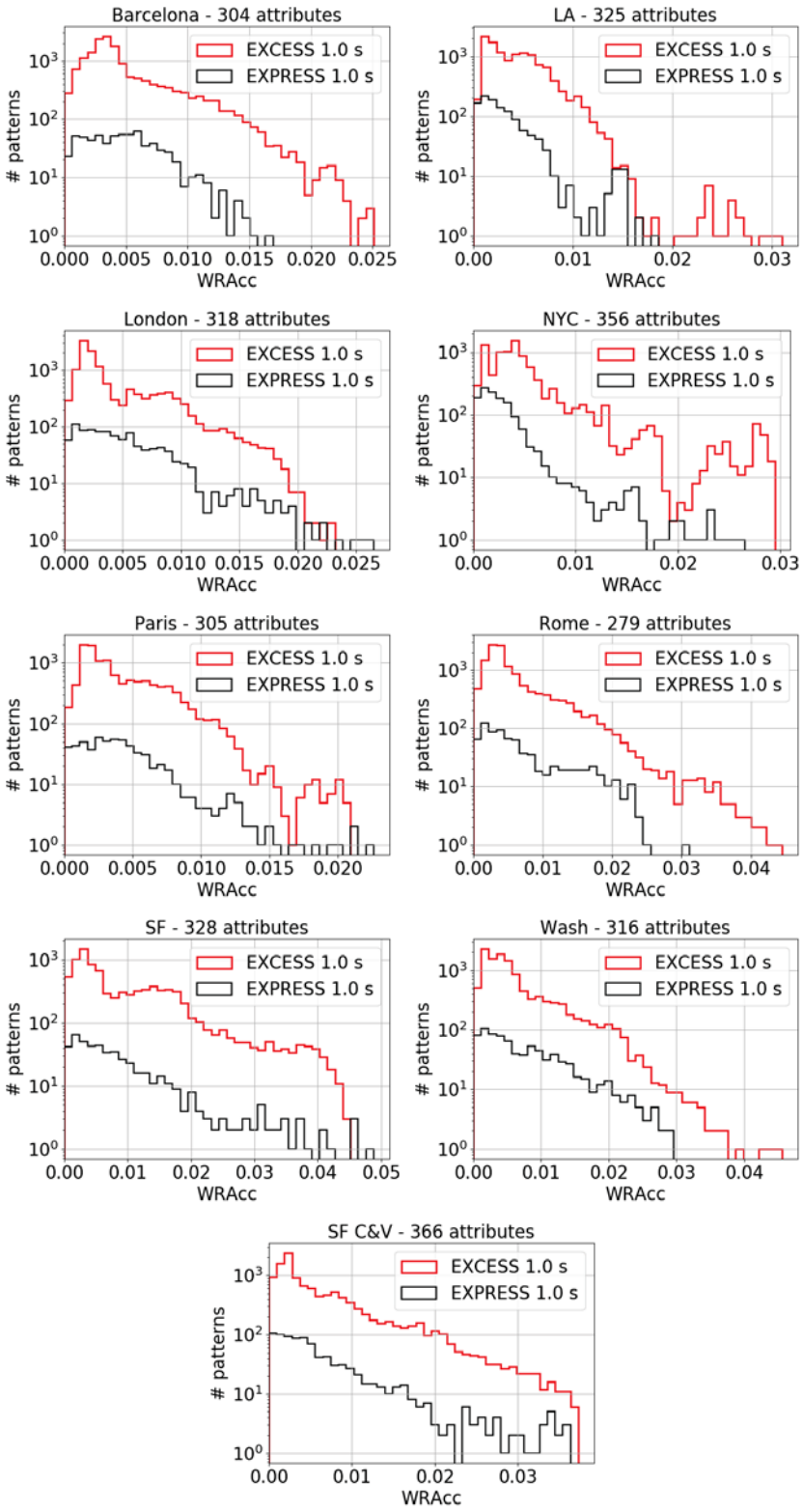
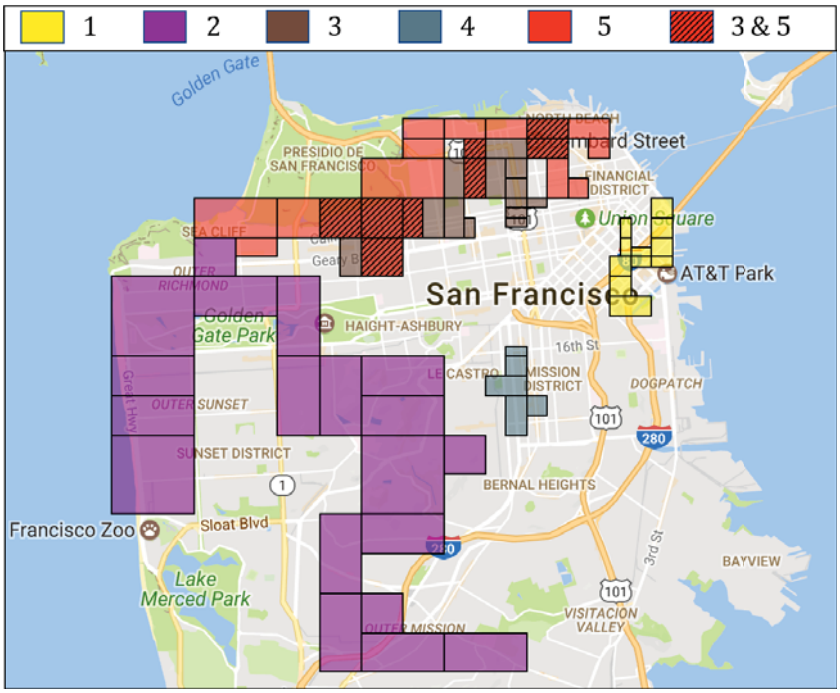


Figure 3.10: Comparison of distributions of patterns sampled with EXCESS and EXPRESS

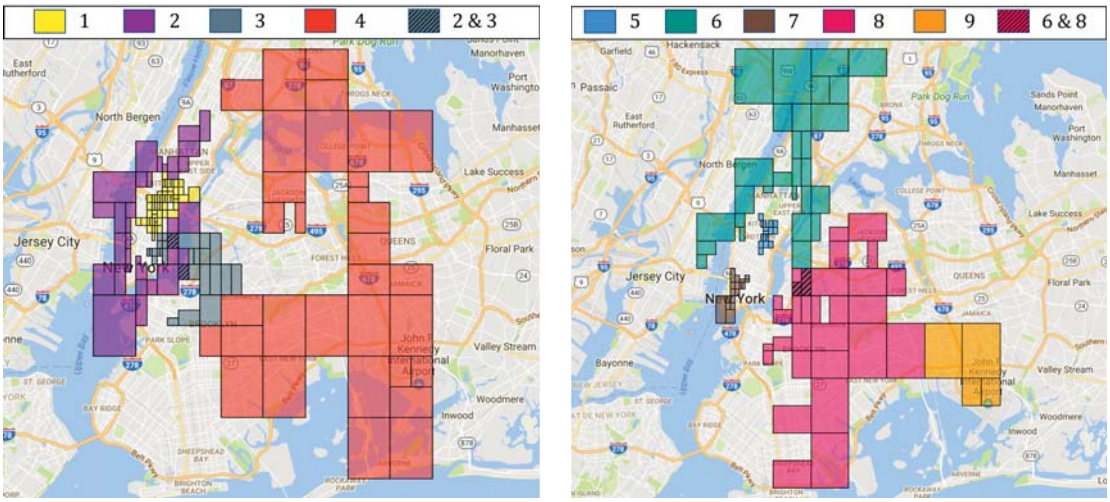
3.4.3 Qualitative study



Id	S^+ Venues	S^+ Crimes	$WRAcc$	Size
P_1	Professional & other places	Embezzlement	0.007	11
P_2	Residence	Vandalism, Vehicle theft, Burglary	0.006	19
P_3	Shop & services	Burglary	0.06	21
P_4	Shop & services, Nightlife spot, Art & Entertainment, Food	Drunkenness	0.003	5
P_5	Outdoors & recreation	Burglary	0.002	20

Figure 3.11: Patterns discovered in San Francisco crimes and venues dataset (49 attributes)

We use CEnergetics on San Francisco crimes and venues dataset to automatically identify typical areas of this city. Fig. 3.11 displays 5 discovered patterns. Pattern P_1 depicts neighbourhoods with a high concentration of venues of type professional & other places, and crimes of type embezzlement. This can be explained by its proximity to the Financial District. P_2 is an area located in the West and South-West of San Francisco. It contains a positive contrast of residences and crimes of type vandalism, vehicle theft, burglary. These crimes are known as the most common types of crimes in residential areas. P_3 and P_5 are overlapping patterns located in the North of the city. They characterize areas with a high concentration of venues of type shop & services, outdoors & recreation, and crimes of type burglary. Pattern P_4 describes an area with a positive contrast of crimes related to drunkenness, which can be explained by the high concentration of nightlife spots in this area.



Id	S^+	S^-	$WRAcc$	size
P_1	Professional & other places	Residence	0.039	50
P_2	Outdoors & recreation	Shop & services, Food	0.023	35
P_3	Nightlife spot	College & university, Travel & transport, Professional & other places	0.02	27
P_4	Travel & transport	Art & entertainment	0.016	32
P_5	Office, Building, Medical center		0.01	16
P_6	Residential building (apartment / condo), Home (private)		0.008	30
P_7	Office		0.008	17
P_8	Home (private), Food & drink shop		0.007	30
P_9	Taxi, Parking, Donut shop, Airport, General travel		0.006	4

Figure 3.12: Patterns discovered in New York datasets with 10 attributes (patterns P_1 to P_4 plotted on the left-hand side map) 356 attributes (patterns P_5 to P_9 plotted on the right-hand side map).

We also report 9 discovered patterns on New York venues dataset. They are presented in Fig. 3.12. Four of them (on the left-hand side map) are discovered on the dataset with 10 attributes, whereas the 5 remaining ones (on the right-hand side map) are discovered on the dataset with 356 attributes. P_1 is located in the South of Central Park. This area is known to be a business and professional area with a low concentration of residences. A sub-area of P_1 is depicted by P_5 with a high concentration of offices, buildings, medical centers. P_2 describes areas with a high proportion of venues of type outdoors & recreation. It contains Central Park and some areas located near East River and Hudson River. P_3 covers a part of the South of Manhattan and the North of Brooklyn, with a high concentration of nightlife spots. P_4 covers John F. Kennedy and LaGuardia Airports and their surroundings. This explains the high

presence of travel & transport venues. More precisely, P_9 contains districts arround John F. Kennedy Airport, and it depicts them with venues of types: Taxi, parkings, donut shops, airport, and general travels. Both P_6 and P_8 represents areas with high proportion of residences. P_8 is also characterized with an important concentration of food & drink shops. P_7 is another pattern that describes a part of South Manhattan with a high concentration of offices.

Besides, we mined exceptional subgraphs on the different cities. In most of them (e.g., Barcelona, Paris, Rome, Los Angeles, London), the nightlife spots are mainly located in the city center. The higher concentration of outdoor & recreation places is surrounding for London. For seaside towns, they are concentrated on the coasts.

3.5 Conclusion

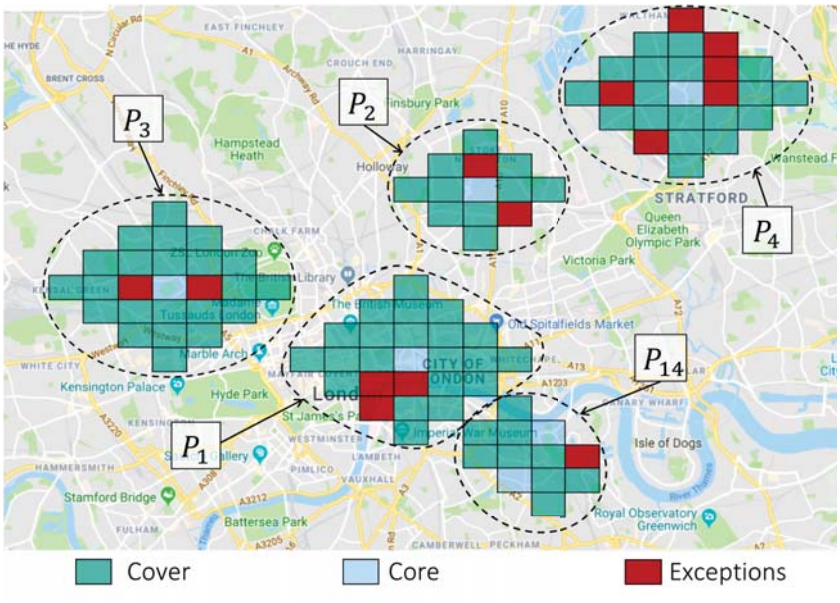
In this chapter, we introduced the exceptional subgraph mining problem in vertex-attributed graph, to discover homogeneous subgraphs that differ from the rest of the graph. While most of local pattern mining methods on vertex-attributed graphs focus their attention on the similarity inside subgraph patterns, the problem defined in this chapter aims to also explicitly consider the exceptionality of the pattern. We defined an efficient algorithm that computes the complete set of exceptional subgraphs by taking advantage of a closure operator, a tight upper bound and other pruning properties. Focusing on closed patterns reduces redundancy among exceptional subgraphs. We also designed an algorithm to sample the output space of closed patterns to enable time-budget analysis. We reported an extensive empirical study over 10 real-world datasets that demonstrates the relevancy of our proposal.

Yet, the proposed approach has several limitations that can be illustrated using patterns shown in qualitative experiments. First, the model used to assess the relevance of patterns does not take into account the user. Even if a pattern depicts some discriminative features, it may be deemed uninteresting if it is expected by her. In Fig. 3.12, a New Yorker may already expect that around Central Parc there is a lot of professional venues, which makes P_1 uninformative in this case. Second, some patterns may be more difficult to assimilate than other ones. One would prefer patterns that don't require a huge effort to be interpreted. In Fig. 3.12, P_1 may be much simpler to grasp than P_4. In fact, P_1 can be simply described as"around Central Parc", as P_1 corresponds to a cohesive set of vertices around this location. However, P_4 is geographically scaterred. The proposed approach ignores the assimilation cost of exceptional patterns. Third, when ranking the results, the approach presented in this chapter does not consider the exceptional subgraphs already communicated to the user. Thus, the top patterns returned may be highly overlapping and give redundant information. In order to fix these issues, the problem of mining exceptional attributed subgraphs is reconsidered in Chapter 4 using the subjective interestingness framework defined by De Bie [59]. As we will see, this framework proposes principled strategies to address the limitations that we have pointed out for the approach designed in this chapter.

CHAPTER 4

Integrating Priors in Attributed Subgraph Mining

Contents



	Characteristics	
ID	**Exceptionally prevalent**	**Exceptionally non-prevalent**
P_1	nightlife, professional	
P_2	nightlife, food	college
P_3	food	
P_4	food	college
P_{14}	professional	college, event

Figure 4.1: Top patterns discovered in the London graph by SIAS-Miner. Green blocks are vertices in the *cover*, blue blocks are *core vertices* that are in the cover, red blocks are *exceptions* (in the neighborhoods but not in the cover). The covers of the top 4 patterns are defined in terms of a single neighborhood with a maximal geodesic radius between two and three, while the cover of the pattern P_{14} is defined as the intersection of two such neighborhoods.

4.1 Introduction

In this chapter, we consider again the problem of mining exceptional attributed subgraphs with, as objective, to address the limitations of the previously proposed solutions. To this end, we introduce a new method, called SIAS-Miner, which exploits an improved pattern syntax and a new interestingness model to measure the quality of patterns. This model is rooted on the subjective interestingness framework proposed in [59] and built upon concepts from Information Theory [58]. SIAS-Miner overcomes the limitations of CEnergetics by: (1) taking into account the background knowledge of the user to provide patterns that are surprising to her (informative), (2) integrating the complexity of patterns to foster the ones that are easy to assimilate (cohesive), (3) updating the interestingness model each time a pattern is displayed in order to continuously return patterns that provide new information. SIAS-Miner allows one to find so-called Cohesive Subgraphs with Exceptional Attributes (*CSEA patterns*), which are easy-to-understand subgraphs across which a specified subset of the attributes consistently have exceptionally

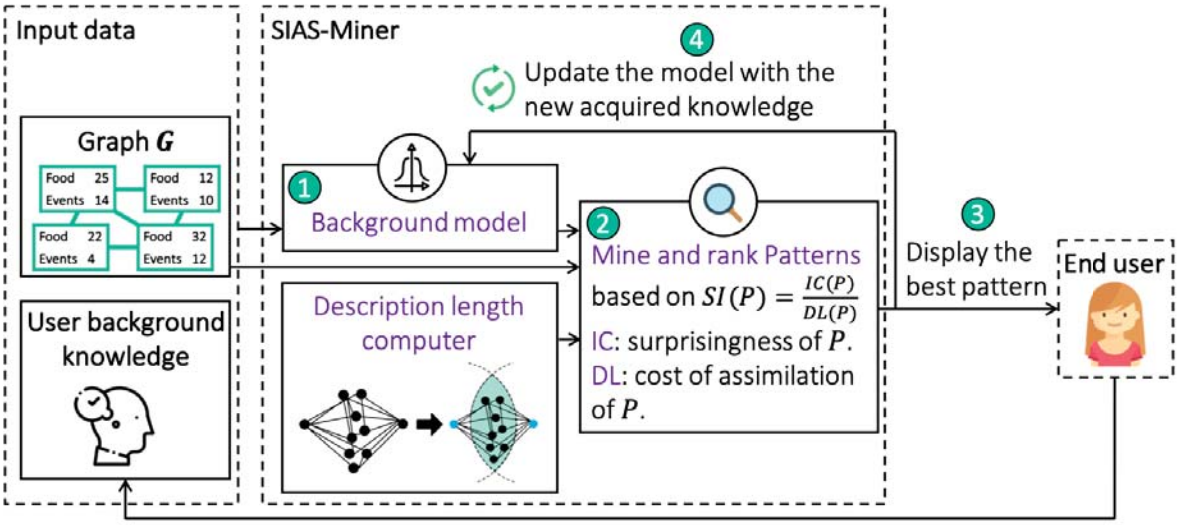


Figure 4.2: Overview of SIAS-Miner.

large or small values.

Fig. 4.1 shows some example CSEA patterns returned by SIAS-Miner, applied to a geographic network of London's districts, where attributes indicate the prevalence of different kinds of venues. CSEA patterns in this London dataset specify some types of venues as particularly prevalent or non-prevalent within all districts in an area. We refer to the subgraph of a CSEA pattern as its *cover*. In a CSEA pattern, the cover is described as the intersection of a set of *neighborhoods* of one or more *core vertices* (shown in blue in Fig. 4.1), with potentially some *exceptions* (vertices within that intersection that are not part of the cover, shown in red in Fig. 4.1). The sizes (geodesic radii) of these neighborhoods can vary and are part of the pattern specification. We refer to the set of exceptional attributes in a CSEA pattern, and their ranges within the cover, as the *characteristic* of the CSEA pattern.

SIAS-Miner is innovative in using a new flexible interestingness measure for exceptional subgraph patterns based on information theory, using a quantification of informativeness and interpretability. The informativeness of a pattern is a function of the number of vertices in the cover (the more the better), the number of attributes in the characteristic (more is better) and the exceptionality of the values for those attributes (also more is better). The exceptionality is quantified with respect to specified background knowledge available about the graph, making the informativeness a subjective measure. The interpretability is quantified in terms of the complexity of communicating a pattern, more specifically its description length. The proposed interestingness measure is the ratio of the informativeness and description length, thus representing the information density within the pattern.

Fig. 4.2 illustrates the main steps of the proposed method. (1) SIAS-Miner derives the background model that represents the user beliefs about the input graph as probability distributions. (2) It mines and ranks the patterns based on their interestingness SI which is the ratio between the information content IC and the assimilation cost DL (description length). The information content of a pattern is

evaluated thanks to the background model. A pattern with a high information gain may involve many vertices. Providing the complete cover to the user would lead to a high assimilation cost and thus a high description length. This is why we aim to provide alternative description of a set of vertices which is easier to assimilate – in the sense that it involves less vertices. This alternative description uses core vertices and radius to depict a set of vertices as those that are at the intersection of the *neighborhood – according to the radius – of the core vertices.* (3) The best pattern P is displayed to the user and (4) the background model is updated in order to consider P as known by the user. Then, these four steps can be repeated as many times as wanted to get in each iteration the best pattern considering the updated background model.

This chapter is organized as follows. We present the CSEA pattern syntax in Section 4.2. We formalize their subjective interestingness in Section 4.3. We explain how to iteratively update the background knowledge during the extraction process in Section 4.4. We study how to mine such subgraphs efficiently in Section 4.5. In Section 4.6, we provide a thorough empirical study on four types of data to evaluate (1) the relevance of the subjective interestingness measure compared to state-of-the-art methods, and (2) the efficiency of the algorithms. We present our conclusions in Section 4.7.

4.2 Cohesive subgraphs with exceptional attributes

In this section we introduce the pattern syntax (the abstract form of the patterns) and argue why patterns of this form are both informative and easy to understand. First, we establish the required notation.

Notation. We assume given a set of vertices V, a set of edges $E \subseteq V \times V$, and a set of numerical attributes on vertices $\hat{A}$ (formally, functions mapping a vertex onto an attribute value), with $\hat{a}(v) \in \mathrm{Dom}_a$ denoting the value of attribute $\hat{a} \in \hat{A}$ on $v \in V$. We denote an *attributed graph* as $G = (V, E, \hat{A})$.

We use hats in $\hat{a}$ and $\hat{A}$ to signify the empirical values of the attributes, whereas a and A denote (possibly random) variables over the same domains. In other terms, for $a \in A$ and $v \in V$, $a(v)$ is a random variable having the empirical value $\hat{a}(v)$ that effectively happens in G. The user is assumed to know the set of vertices and the connection structure, so V and E always correspond to the actual graph structure.

With $N_d(v)$ we denote the neighborhood of range d of a vertex v, i.e., the set of vertices whose graph geodesic distance (the number of edges in a shortest path connecting them) to v is at most d:

$$N_d(v) = \{u \in V \mid dist(v, u) \leq d\}.$$

Pattern definition. As described in the introduction, we are interested in patterns that inform the user that a set of attributes has exceptional values within a cohesive set of vertices in the graph. To this end, we propose the following syntax.

Definition 4.1. A *cohesive subgraph with exceptional attributes (CSEA) pattern* consists of a tuple (U, S), where $U \subseteq V$ is a set of vertices in the graph that we refer to as the *cover*, and S is a *characteristic* of these vertices, that is to say S is made of restrictions on the value domains of some attributes of A.

More specifically, $S \subseteq \{(a, [k_a, \ell_a]) \mid a \in A\}$. Furthermore, to be a CSEA pattern, (U, S) has to be contained in G, i.e.

$$\forall (a, [k_a, \ell_a]) \in S \text{ and } \forall u \in U,\ k_a \leq \hat{a}(u) \leq \ell_a.$$

Putting this in words, for every vertex in the cover $u \in U$, the empirical value $\hat{a}(u)$ for attribute a falls within the interval $[k_a, \ell_a]$ specified as a part of the characteristic for the CSEA pattern (U, S).

Notice that U is not restricted to have any particular structure. Rather, cohesiveness will be promoted through the definition of a description length quantifying the complexity to communicate a particular set of vertices to the user. This is explained below in Section 4.3.3.

Intuition behind quantifying the interestingness of CSEA patterns. Informally speaking, a CSEA pattern is more informative if the ranges in S are smaller, as then it conveys more information to the data analyst. This can be formalized as a partial order relation over the characteristics.

Definition 4.2. The partial order relation $\preceq$ between two characteristics S and S' is:

$$S \preceq S' \Leftrightarrow \forall (a, [k'_a, l'_a]) \in S' : \exists (a, [k_a, l_a]) \in S \text{ with } [k_a, l_a] \subseteq [k'_a, l'_a].$$

A 'smaller' characteristic in this partial order is more specific and thus more informative. We will make this more formal in Section 4.3.1, and later use this to efficiently mine informative patterns.

Fig. 4.3 shows a toy graph where vertices correspond to geographical areas described by number of different venues. An edge links vertices that correspond to adjacent areas (that share a part of their borders). A pattern (U, S) that can be interesting is: $(U = \{v_3, v_7\}, S = \{(food, [30, 32]), (college, [0, 1])\})$. Indeed, vertices in U contain a higher (resp. lower) number of food (resp. college) venues comparing with the rest of the graph, and their numbers fall within the intervals specified in S.

At the same time, a CSEA pattern (U, S) is more interesting if we can describe the cover U more concisely in some *intuitive description*. Thus, along with the pattern syntax, we must also specify how a pattern from this language will be described. To this end, we propose to describe the cover U as a neighborhood of a specified range from a given specified vertex, or more generally as the intersection of a set of such neighborhoods. For enhanced expressive power, we additionally allow for the description to specify exceptions on the above: vertices that do fall within this (intersection of) neighborhood(s), but which are to be excluded from the cover U, because they do not exhibit the same characteristic. Exceptions increase the complexity to interpret such patterns (as will be quantified in the description length), but greatly increase the expressive power of the CSEA pattern syntax.

A premise of this work is that this way of describing the set U is intuitive for human analysts, such that the length of the description of a pattern, as discussed in detail in Section 4.3.3, is a good measure of the complexity to assimilate or understand it. The qualitative experiments reported in Section 4.6 appear to confirm that this is the case.

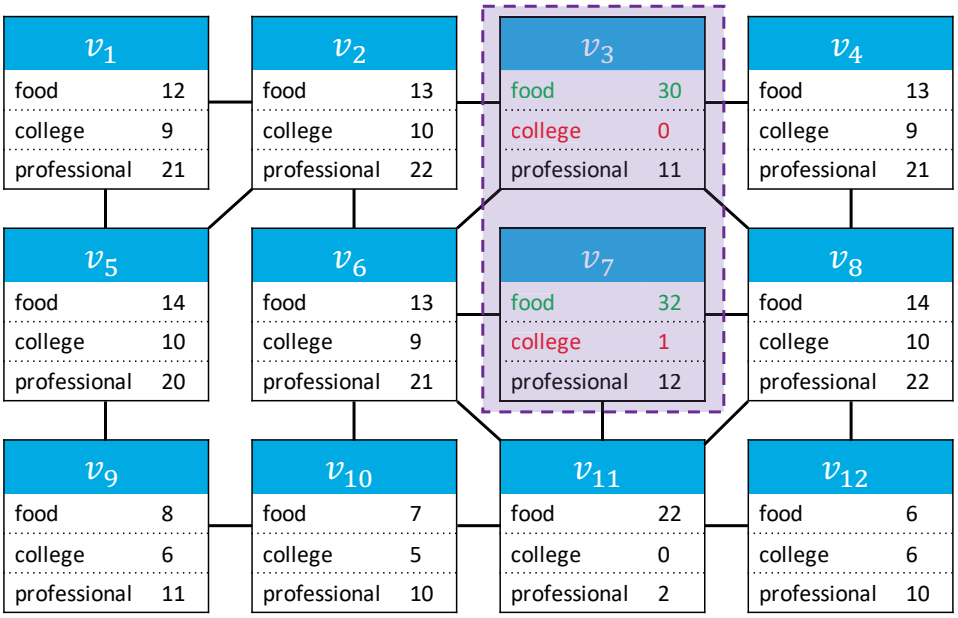


Figure 4.3: A toy graph: vertices are geographical regions and they are described by number of different types of venues. Edges connect the vertices that correspond to neighbouring regions.

4.3 Subjective interestingness of CSEA patterns

The previous sections already hinted at the fact that we will formalize the interestingness of a CSEA pattern (U, S) by trading off the amount of information contained in the pattern against the complexity of interpreting the pattern. We will use information theory to quantify both the informativeness and the complexity using the strategy outlined in [59][1].

Precise definitions will be given below, but first we introduce the statistic ultimately used to rank patterns. The Information Content (IC) of a CSEA pattern, which quantifies the amount of information contained in a pattern, depends on both the cover U and the characteristic S. Intuitively, it should be larger when more vertices are involved, when the intervals are narrower, and when they are more extreme. We denote the information content as $\mathrm{IC}(U, S)$.

The Description Length (DL), which quantifies the interpretation complexity, also depends on U and S and will be denoted as $\mathrm{DL}(U, S)$. Likewise, communicating larger characteristics is strictly more time consuming, but we will not describe U directly, so the DL for a set U is more intricate as discussed in Section 4.3.3. We will rank patterns by the quantity that we call the Subjective Interestingness (SI) of a CSEA pattern (U, S) defined as follows.

Definition 4.3. The Subjective Interestingness of a CSEA pattern (U, S) is the ratio between the information content $\mathrm{IC}(U, S)$ and the description length $\mathrm{DL}(U, S)$:

$$\mathrm{SI}(U, S) = \frac{\mathrm{IC}(U, S)}{\mathrm{DL}(U, S)}.$$

[1] This approach is now known as the FORSIED framework.

4.3.1 The information content of a CSEA pattern

The information carried by a pattern is quantified by the information content [59], a quantity also known as the self-information or surprisal [58].

Definition 4.4. The information content of a pattern (U, S) equals the reduction in uncertainty about the data when we learn about the pattern, and is defined as

$$\mathrm{IC}(U, S) = -\log(\mathrm{Pr}(U, S)).$$

Here $\mathrm{Pr}(U, S)$ is the probability that the CSEA (U, S) is present in the data. This explicates that we have to define such a distribution over the space of all patterns. We can achieve this as follows. The data $\hat{\omega}$ can be seen as a sample from the space of all possible vertex-attributed graphs Ω, where $G = (V, E, \hat{A}) = \hat{\omega} \in \Omega$ and all elements in Ω have the same set of vertices and edges, since we assume these are known, while the attribute values are unknown to the user. Then let Pr denote a probability distribution over the set Ω of possible vertex-attributed graphs with vertices V and edges E (i.e., the possible value combinations of A). We refer to Pr as the *background distribution* and will introduce a convenient and tractable choice in the following section.

Generically, given a distribution over all possible datasets, we can obtain a distribution over patterns, by observing that a pattern is a set $\Omega' \subseteq \Omega$ that specifies $\hat{\omega}$ falls within Ω' and not outside of Ω' [59, 140]. I.e., it may reduce the value combinations deemed possible and hence provide information. The probability of a pattern Ω' can then be computed through integration, i.e., $\mathrm{Pr}(\Omega') = \int_{\omega \in \Omega'} \mathrm{Pr}(\omega) \mathrm{d}\omega$.

In this context, $\Omega' = (U, S)$, which limits the possible attribute values of the vertices in the cover U. How to compute the probability of a CSEA pattern (U, S) is considered in more detail in the following sections.

The power of this approach is that we quantify the IC of a pattern against a prior belief state about the data. It rigorously models the fact that the more plausible the pattern is according to a model, the less information a pattern provides, and thus the smaller the information content ought to be. It is possible to specify the model accounting for (user specific) background knowledge and hence affect the ranking of patterns in a subjective manner.

In Section 4.3.2, we first discuss which prior beliefs could be appropriate for CSEA patterns, and how to infer the corresponding background distribution. Then, in Section 4.3.3, we discuss how the description length $\mathrm{DL}(U, S)$ can be defined appropriately.

4.3.2 Information content and prior beliefs for count attributes

Positive integers as attributes. For concreteness, let us consider the situation where the attributes are positive integers $(a : V \to \mathbb{N}, \forall a \in A)$, as will be our main focus throughout this work[2]. For example, if the vertices are geographical regions (with edges connecting vertices of neighboring regions), then

[2]The results presented can be extended relatively directly for boolean and real-valued attributes. [60] shows how to derive the background distribution $\mathrm{Pr}(A)$ corresponding to these types of attributes.

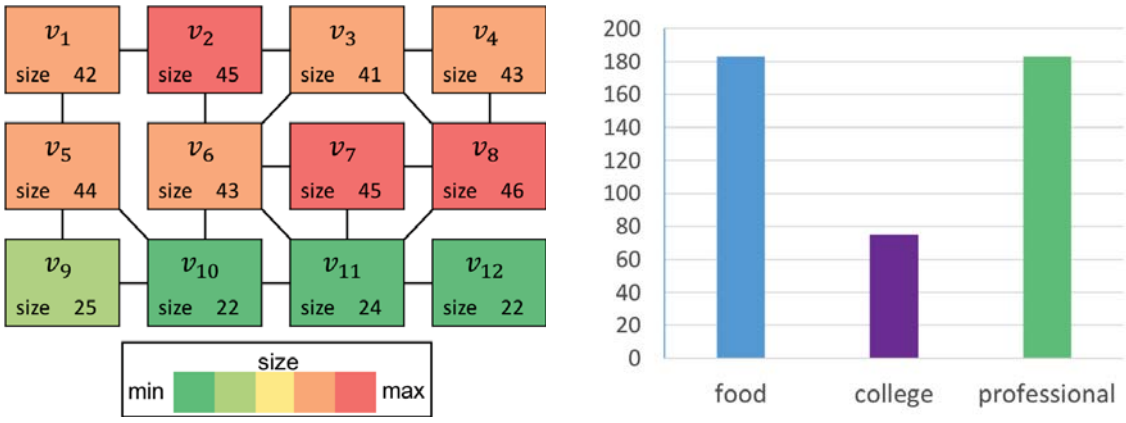


Figure 4.4: Statistics corresponding to the formalized constraints for the case of the toy graph of Fig. 4.3. Left: The distribution of vertices sizes (the first constraint). Right: The total number of each venue type (the second constraint).

the attributes could be counts of particular types of places in the region (e.g. one attribute could be the number of shops). It is clear that it is less informative to know that an attribute value is large in a large region than it would be in a small region. Similarly, a large value for an attribute that is generally large is less informative than if it were generally small. The above is only true, however, if the user knows (or believes) *a priori* at least approximately what these averages are for each attribute, and what the 'size' is of each region. Such prior beliefs can be formalized as equality constraints on the values of the attributes A on all vertices, or mathematically:

$$\sum_A \Pr(A) \left(\sum_{a \in A} a(v) \right) = \sum_{\hat{a} \in \hat{A}} \hat{a}(v), \quad \forall v \in V, \tag{4.1}$$

$$\sum_A \Pr(A) \left(\sum_{v \in V} a(v) \right) = \sum_{v \in V} \hat{a}(v), \quad \forall a \in A. \tag{4.2}$$

The first constraint means that the user already knows the size (the total count) of each vertex, while the second constraint means that the user knows the total count of each attribute in the overal graph. Even if the user does not have these priors, they also can be easily communicated to her before the mining process through simple statistical tools as shown in Fig. 4.4 for the toy graph.

These constraints will not be sufficient to uniquely determine the distribution Pr(A). A common strategy to overcome this problem is to search for the distribution that has the largest entropy subject to these constraints, to which we will refer as the MaxEnt distribution. The argument of this choice is that any distribution other than the MaxEnt distribution effectively makes additional assumptions about the data that reduce the entropy. As making additional assumptions biases the distribution, the MaxEnt distribution is the most rigorous choice.

The MaxEnt background distribution can then be found as the probability distribution Pr maximizing the entropy $-\sum_A \Pr(A) \log \Pr(A)$, subject to these constraints (in Equations 4.1 and 4.2) and the

normalization $\sum_A \Pr(A) = 1$. As shown in [60], the optimal solution of this optimization problem is a product of independent Geometric distributions, one for each vertex attribute-value $a(v)$. Each of these Geometric distributions is of the form $\Pr(a(v) = z) = p_{av} \cdot (1 - p_{av})^z$, $z \in \mathbb{N}$, where p_{av} is the success probability and it is given by: $p_{av} = 1 - exp(\lambda_a^r + \lambda_v^c)$, with λ_a^r and λ_v^c the Lagrange multipliers corresponding to the two constraint types. The optimal values of these multipliers can be found by solving the convex Lagrange dual optimization problem.

Given these Geometric distributions for the attribute values under the background distribution, we can now compute the probability of a pattern (U, S) as follows:

$$\begin{aligned}\Pr(U, S) &= \prod_{v \in U} \prod_{(a,[k_a,\ell_a]) \in S} \Pr(a(v) \in [k_a, \ell_a]), \\ &= \prod_{v \in U} \prod_{(a,[k_a,\ell_a]) \in S} \left((1 - p_{av})^{\lceil k_a \rceil} - (1 - p_{av})^{\lfloor \ell_a \rfloor + 1} \right).\end{aligned}$$

This can be used directly to compute the information content of a pattern on given data, as the negative log of this probability. However, the pattern syntax is not directly suited to be applied to count data, when different vertices have strongly differing total counts. The reason is that the interval of each attribute is the same across vertices, which is desirable to keep the syntax understandable. Yet, if neighboring regions have very different total counts, the same interval could be very informative to some vertices while being uninformative to others, which makes it hard to find CSEA patterns with a characteristic that is informative for all vertices in its cover.

Let us illustrate this issue with an example from Fig. 4.3. The graph contains the pattern $(U = \{v_3, v_7\}, S = \{(food, [30, 32]), (college, [0, 1])\})$. This can be interpreted as a relatively high presence of food venues and low presence of college. Likewise in v_{11}, there is no college and the number of food venues is significantly high. Even if $\hat{food}(v_{11})$ is only 22, this still makes sens because the size of v_{11} is only 24, while the size of v_3 and v_7 is more than 40. We would like to inform the user that these same prevalences about food and college are present in all of $\{v_3, v_7, v_{11}\}$. However, v_{11} does not contain $S = \{(food, [30, 32]), (college, [0, 1])\}$. In order to contain all $\{v_3, v_7, v_{11}\}$, we need to use the restriction $(food, [22, 32])$ instead. This restriction is larger and much less informative than $(food, [30, 32])$, especially for v_3 and v_7 which are vertices with great sizes. We need a different way that allows to take into account the size of each vertex when establishing the characteristic.

p-values as attributes. To address this problem, we propose to search for the patterns not on the counts themselves, but rather on their *significance* (i.e., p-value or tail probability), computed with the background distribution as null hypothesis in a one-sided test. More specifically, we define the quantities $\hat{c}_a(v)$ as

$$\begin{aligned}\hat{c}_a(v) &\triangleq \Pr(a(v) \geq \hat{a}(v)), \\ &= (1 - p_{av})^{\hat{a}(v)},\end{aligned}$$

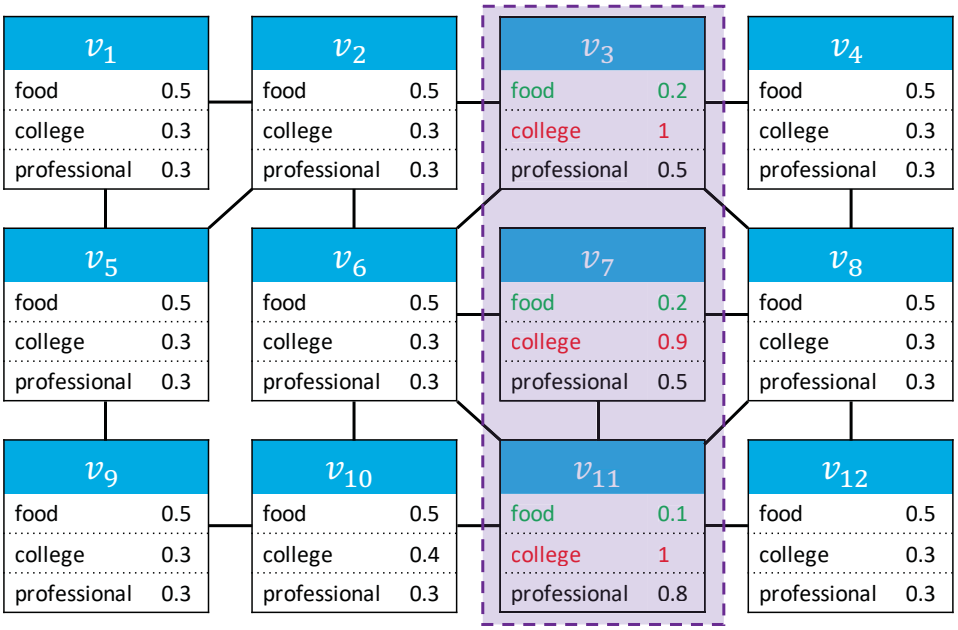


Figure 4.5: The toy graph of Fig. 4.3 with normalized attribute values (p-values indicating how over-represented the value is as compared to a MaxEnt model with category and vertex size constraints, see Section 4.3.2).

and use this instead of the original attributes $\hat{a}(v)$. This transformation of $\hat{a}(v)$ to $\hat{c}_a(v)$ can be regarded as a principled normalization of the attribute values to make them comparable across vertices.

In other words, $\hat{c}_a(v)$ is the probability that the expected value of $a(v)$ by the user is higher than the observed value $\hat{a}(v)$. Low values of $\hat{c}_a(v)$ correspond to over-expressed attributes, because it means that the user does not expect a value of $a(v)$ as large as the observed one $\hat{a}(v)$, while high values of $\hat{c}_a(v)$ correspond to under-expressed attributes.

For example, after applying this transformation to the toy graph of Fig. 4.3, this gives the p-values presented in Fig. 4.5. Let us take the vertex v_3 and the attribute $food$, the transformation gives $\hat{c}_{food}(v_3) \triangleq \Pr(food(v_3) \geq 30) = 0.2$. An example of pattern (U, S) is such that $U = \{v_3, v_7, v_{11}\}$ and $S = \{(food, [0, 0.2]), (college, [0.9, 1])\}$ (an over-expression of food and an under expression of college). Notice that even if the size of v_{11} (and consequently the number of its food venues) is lower than the sizes of v_3 and v_7, the p-value normalization made it possible to capture the common characteristic S that covers all the set $U = \{v_3, v_7, v_{11}\}$.

To compute the IC of a pattern with the transformed attributes $\hat{c}_a$, we must be able to evaluate the probability that $c_a(v)$ falls within a specified interval $[k_{c_a}, \ell_{c_a}]$ under the background distribution for $a(v)$. In other terms, what is the probability that the significance of $\hat{a}(v)$ falls within $[k_{c_a}, \ell_{c_a}]$? How

much surprising is it? This is given by:

$$\begin{aligned}
&\Pr(c_a(v) \in [k_{c_a}, \ell_{c_a}]) = \Pr\Big(\Pr(a(v) \geq \hat{a}(v)) \in [k_{c_a}, \ell_{c_a}]\Big), \\
&= \Pr\left((1-p_{av})^{a(v)} \in [k_{c_a}, \ell_{c_a}]\right), \\
&= \Pr\left(a(v) \leq \frac{\log(k_{c_a})}{\log(1-p_{av})} \wedge a(v) \geq \frac{\log(\ell_{c_a})}{\log(1-p_{av})}\right), \\
&= \Pr\left(a(v) \leq \left\lfloor \log_{1-p_{av}}(k_{c_a}) \right\rfloor \wedge a(v) \geq \left\lceil \log_{1-p_{av}}(\ell_{c_a}) \right\rceil\right), \\
&= \Pr\left(a(v) \geq \left\lceil \log_{1-p_{av}}(\ell_{c_a}) \right\rceil\right) - \Pr\left(a(v) \geq \left\lfloor \log_{1-p_{av}}(k_{c_a}) \right\rfloor + 1\right), \\
&= (1-p_{av})^{\left\lceil \log_{1-p_{av}}(\ell_{c_a}) \right\rceil} - (1-p_{av})^{\left\lfloor \log_{1-p_{av}}(k_{c_a}) \right\rfloor + 1}.
\end{aligned}$$

The last line of the equation only depends on three given values: p_{av}, k_{c_a}, ℓ_{c_a}. In order to simplify the notations, let us define a function $\rho : (0,1]^3 \longrightarrow (0,1]$, $\rho(x,y,z) = (1-x)^{\left\lceil \log_{1-x}(z) \right\rceil} - (1-x)^{\left\lfloor \log_{1-x}(y) \right\rfloor + 1}$. In what follows, we will use ρ to express the latter probability as:

$$\Pr(c_a(v) \in [k_{c_a}, \ell_{c_a}]) = \rho(p_{av}, k_{c_a}, \ell_{c_a}).$$

Thus, the IC of a pattern on the transformed attributes $\hat{c}$ can be calculated as:

$$\begin{aligned}
&\mathrm{IC}(U,S) = -\log(\Pr(U,S)), \\
&= -\sum_{(a,[k_{c_a},\ell_{c_a}])\in S} \sum_{v \in U} \log(\rho(p_{av}, k_{c_a}, \ell_{c_a})).
\end{aligned} \tag{4.3}$$

In this work, we focus on intervals $[k_{c_a}, \ell_{c_a}]$ where either $k_{c_a} = 0$ (the minimal value) and $\ell_{c_a} < 0.5$, or $\ell_{c_a} = 1$ (the maximal value) and $k_{c_a} > 0.5$. Such intervals state that the values of an attribute are all significantly large[3] or significantly small respectively, for all vertices in U. We argue such intervals are easiest to interpret.

4.3.3 Description length

Definition 4.5. The description length measures the complexity of communicating a pattern (U,S) to the user. It can be defined as the complexity of communicating U and S:

$$\mathrm{DL}(U,S) = \mathrm{DL}_A(S) + \mathrm{DL}_V(U),$$

where $\mathrm{DL}_A(S)$ (resp. $\mathrm{DL}_V(U)$) is the description length of S (resp. U).

[3] Note empty regions have tail probabilities $\hat{c}_a(v) = 1$ for any attribute and thus fall within any upper interval, but also IC $= 0$ for any attribute of that region as both $l_{c_a} = 1$ and $p_{av} = 1$.

Description length of attributes $\mathrm{DL}_A(S)$: The higher the number of attributes in S, the harder its communication to the user could be. This can be suitably represented by:

$$\mathrm{DL}_A(S) = (|S| + 1) \cdot \log(|A|) + \sum_{(a,[k_{c_a},\ell_{c_a}])\in S} (1 + \log(M_a)),$$

with $M_a = |\{\hat{a}(v) \mid v \in V\}|$, the number of distinct values of $\hat{a}$ on the graph. More precisely, the first term accounts for the encoding of the attributes that are restricted. Encoding an attribute over $|A|$ possibilities costs $\log(|A|)$ bits. We do this encoding $(|S| + 1)$ times, one for each attribute in S plus one for the length of S. The second term is the length of the encoding of restriction $(a, [k_{c_a}, \ell_{c_a}]) \in S$. One bit is used to specify the type of interval ($[0, x]$ or $[x, 1]$) and the encoding of the other bound of the interval is in logarithm of the number of distinct values of a on the graph.

Description length of vertices $\mathrm{DL}_V(U)$: As mentioned above, we describe the vertex set U in the pattern as (the intersection of) a set of neighborhoods $N_d(v),\ v \in V$, with a set of exceptions: vertices that are in the intersection but not part of the cover U. The length of such a description is the sum of the description lengths of the neighborhoods and the exceptions. More formally, let us define the set of all neighborhoods $\mathcal{N} = \{N_d(v) \mid v \in V \wedge d \in \mathbb{N} \wedge d \leq D_N\}$ (with D_N the maximum range d considered), and let $\mathcal{N}(U) = \{N_d(v) \in \mathcal{N} \mid U \subseteq N_d(v)\}$ be the subset of neighborhoods that contain U. The length of a description of the set U as the intersection of all neighborhoods in a subset $X \subseteq \mathcal{N}(U)$, along with the set of exceptions $\mathrm{exc}(X, U) \triangleq \cap_{N_d(v)\in X} N_d(v) \setminus U$, is then quantified by the function $\mathrm{desc} : 2^{\mathcal{N}(U)} \times U \longrightarrow \mathbb{R}$ defined as:

$$\mathrm{desc}(X, U) = (|X| + 1) \cdot \log(|\mathcal{N}|) + (|\mathrm{exc}(X, U)| + 1) \cdot \log(|\cap_{x\in X} x|).$$

Indeed, the first term accounts for the description of the number of neighborhoods ($\log(|\mathcal{N}|)$, and for the description of which neighborhoods are involved ($|X| \log(|\mathcal{N}|)$). The second term accounts for the description of the number of exceptions ($\log(|\cap_{x\in X} x|)$), and for the description of the exceptions themselves ($|\mathrm{exc}(X, U)| \log(|\cap_{x\in X} x|)$).

Clearly, there is generally no unique way to describe the set U. The best one is thus the one that minimizes desc. But also, in several applications, we need to limit the number of core vertices used to describe U. This means limiting $|X|$ to some parameter α whose default value is $\alpha = |V|$. This finally leads us to the definition of the description length of U as:

$$\mathrm{DL}_V(U) = \min_{\substack{X\subseteq\mathcal{N}(U)\\ |X|\leq\alpha}} \mathrm{desc}(X, U).$$

In Fig. 4.5, the set of vertices $U = \{v_3, v_7, v_{11}\}$ can be described by $X_1 = \{N_1(v_8)\}$ with three exceptions $\mathrm{exc}(X_1, U) = \{v_8, v_4, v_{12}\}$, and with a length $\mathrm{desc}(X_1, U) = 21.5$. Another possible description of U is $X_2 = \{N_1(v_6), N_1(v_8)\}$ with no exception (since $N_1(v_6) \cap N_1(v_8) = U$) and with a length $\mathrm{desc}(X_2, U) = 18.3$. Based on the values of the function desc, the description X_2 is better than X_1. Among all the possible descriptions of U, it turns out that X_2 is the one that minimizes $\mathrm{desc}(X, U)$, consequently, $\mathrm{DL}_V(U) = \mathrm{desc}(X_2, U)$.

4.4 Iterative mining of CSEA patterns

When a pattern $P_0 = (U_0, S_0)$ is observed by a rational user, her background knowledge will change to take into account this newly learned piece of information. It results that the pattern P_0 becomes expected by her. We need as well to update our interestingness model such that the probability that the data contains the pattern P_0 becomes equal to 1. Also, this model updating will decrease the IC of other patterns if a part of their information overlaps with the one of P_0. Patterns that have a large overlap with P_0 will then no longer be deemed interesting since their information content will substantially decrease. This approach of modifying the interestingness model to account for previously seen patterns is a natural way to avoid presenting multiple redundant patterns to the user.

Let us define $\text{SI}(P \mid P_0)$ the subjective interestingness of a pattern $P = (U, S)$ conditioned on the presence of the already observed pattern P_0:

$$\text{SI}(P \mid P_0) = \frac{\text{IC}(P \mid P_0)}{\text{DL}(P)} = \frac{-\log(\Pr(P \mid P_0))}{\text{DL}(P)}.$$

$\Pr(P \mid P_0)$ is the probability that $P = (S, U)$ is present in the data given that $P_0 = (U_0, S_0)$ is present:

$$\Pr(P \mid P_0) = \prod_{v \in U} \prod_{(a,[k_{c_a}, \ell_{c_a}]) \in S} \Pr(c_a(v) \in [k_a, \ell_a] \mid P_0).$$

The value of $\Pr(c_a(v) \in [k_a, \ell_a] \mid P_0)$ for each pair of attribute $a \in A$ and vertex $v \in V$ can be computed using the law of conditional probability:

$$\Pr(c_a(v) \in [k_a, \ell_a] \mid P_0) = \frac{\Pr(c_a(v) \in [k_a, \ell_a] \wedge P_0)}{\Pr(P_0)}.$$

We consider two cases:

1. If P_0 does not give any information about $c_a(v)$ (the vertex v is not in U_0, or there is no restriction of a in S_0), then the observation of P_0 has no impact on the probability $\Pr(c_a(v) \in [k_{c_a}, \ell_{c_a}] \mid P_0)$:

$$\begin{aligned} \Pr(c_a(v) \in [k_{c_a}, \ell_{c_a}] \mid P_0) &= \Pr(c_a(v) \in [k_{c_a}, \ell_{c_a}]), \\ &= \rho(p_{av}, k_{c_a}, \ell_{c_a}). \end{aligned}$$

2. Otherwise, $v \in U_0$ and S_0 contains a restriction of a, let it be $(a, [k_0, \ell_0])$, then:

$$\begin{aligned} \Pr(c_a(v) \in [k_{c_a}, \ell_{c_a}] \mid P_0) &= \Pr(c_a(v) \in [k_{c_a}, \ell_{c_a}] \mid c_a(v) \in [k_0, \ell_0]), \\ &= \frac{\Pr(c_a(v) \in [k_{c_a}, \ell_{c_a}] \cap [k_0, \ell_0])}{\Pr(c_a(v) \in [k_0, \ell_0])}, \\ &= \frac{\Pr(c_a(v) \in [\max(k_0, k_{c_a}), \min(\ell_0, \ell_{c_a})])}{\Pr(c_a(v) \in [k_0, \ell_0])}, \\ &= \frac{\rho\left(p_{av}, \max(k_0, k_{c_a}), \min(\ell_0, \ell_{c_a})\right)}{\rho(p_{av}, k_0, \ell_0)}. \end{aligned}$$

Notice that in this second case, $[k_0, \ell_0]$ and $[k_{c_a}, \ell_{c_a}]$ necessarily overlap. Otherwise, the empirical value $\hat{c}_a(v)$ needs to belong to two completely disjoint intervals to make P and P_0 hold in G, which is absurd.

As explained in Section 4.5.4, the set of CSEA patterns can be ordered using the IC measure evaluated conditional on the knowledge of the patterns ranked before.

4.5 SIAS-Miner algorithm

SIAS-Miner (Algorithm 3) mines interesting patterns using an enumerate-and-rank approach. First, it enumerates all CSEA patterns (U, S) that are closed simultaneously with respect to U, S, and the neighborhood description. Second, it ranks patterns according to their SI values. The calculation of $\text{IC}(U, S)$ and $\text{DL}_A(S)$ is simple and direct. However, computing $\text{DL}_V(U)$ is not trivial, since there are several ways to describe U and we are looking for the one minimizing $\text{desc}(X, U)$. To achieve this goal, we propose an efficient algorithm DL_V-Optimise that calculates the minimal description of U and stores the result in the mapping structure $minDesc$. This algorithm is presented in Section 4.5.2.

4.5.1 Pattern enumeration

The exploration of the search space is based on subgraph enumeration. We only enumerate sets of vertices $U \subseteq V$ that are covered by a non empty characteristic $S \neq \emptyset$ and that can be described with neighbors $\mathcal{N}(U) \neq \emptyset$. Yet, a subgraph $G[U]$ can be covered by a large number of characteristics S that leads to as many redundant patterns. This can be avoided by only considering the most specific characteristic that covers U, as the other patterns do not bring any additional information.

Definition 4.6. Given a set of vertices $U \subseteq V$, the function $max_S(U)$ returns the most specific characteristic, also made of significant intervals, associated to U:

$$\begin{aligned} max_S(U) = \{(a, [k_{c_a}, \ell_{c_a}]) \mid a \in A \wedge \\ ((k_{c_a} = 0 \wedge \ell_{c_a} = \max_{v \in U} \hat{c}_a(v) \wedge \ell_{c_a} < 0.5) \vee \\ (\ell_{c_a} = 1 \wedge k_{c_a} = \min_{v \in U} \hat{c}_a(v) \wedge k_{c_a} > 0.5))\}. \end{aligned}$$

For example in Fig. 4.3, the set of vertices $U' = \{v_3, v_7\}$ is covered by a large number of characteristics, but there is only one characteristic that is the most specific: $max_S(U') = \{(food, [0, 0.2]), (college, [0.9, 1])\}$. We remind that in this Figure, the attributes are normalized counts (p-values of original counts).

Moreover, it may happen that two sets of vertices U and U', such that $U' \subseteq U$, are covered by the same characteristic ($max_S(U') = max_S(U)$) and are described by the same neighborhoods ($\mathcal{N}(U') = \mathcal{N}(U)$). In that case, U brings more information than U' and its vertex description length $\text{DL}_V(U)$ is lower or equal than $\text{DL}_V(U')$, as all the descriptions $X \subseteq \mathcal{N}(U')$ also cover U with a lower or equal

Algorithm 3: SIAS-Miner (U, C, $Result$, $minDesc$)

Input: U: the current set of enumerated vertices, C: the set of candidates vertices.
Output: $Result$: the set of CSEAs, $minDesc$: a mapping structure that stores the minimum description of U for each pattern (U, S).

```
1  if C ≠ ∅ then
2      // choose a candidate vertex with the fail first principle
3      v ← argmin_{v∈C} |max_S(U ∪ {v})|
4      U' ← clo(U ∪ {v})
5      if U' ⊆ U ∪ C then
6          // We prune candidates that cannot be used anymore
7          C' ← {v ∈ C | max_S(U' ∪ {v}) ≠ ∅ ∧ 𝒩(U' ∪ {v}) ≠ ∅}
8          SIAS-Miner(U', C' \ U', Result, minDesc)
9      SIAS-Miner(U, C \ {v}, Result, minDesc)
10 else
11     Result ← Result ∪ {(U, max_S(U))}
12     bestDesc ← ∅
13     DL_V-Optimise(U, ∅, 𝒩(U), bestDesc)
14     minDesc[(U, max_S(U)] ← bestDesc
```

number of exceptions. Hence, the pattern $(U', max_S(U'))$ is not useful. This motivates the idea of only exploring patterns (U, S) that are closed with respect to both S and $\mathcal{N}(U)$. Closing a set of vertices U w.r.t. S and $\mathcal{N}(U)$ maximizes IC(U), and minimizes DL$_V(U)$. Although this could increase the value of DL$_A(S)$, we believe that this choice is very suitable as it allows to drastically improve the performance of the algorithm and reduce the size of the output, without altering the result quality. We define the closure function $clo : 2^V \longrightarrow 2^V$ which is fundamental for our method.

Definition 4.7. For a given set $U \subseteq V$, $clo(U)$ extends U by adding vertices that keep $max_S(U)$ and $\mathcal{N}(U)$ unchanged:

$$clo(U) = \{v \in V \mid max_S(v) \preceq max_S(U) \wedge \mathcal{N}(U) \subseteq \mathcal{N}(\{v\})\}.$$

$clo(U)$ is indeed a closure function since it is extensive ($U \subseteq clo(U)$), idempotent ($clo(U) = clo(clo(U))$), and monotonic (if $X \subseteq Y$, then $clo(X) \subseteq clo(Y)$). In Fig. 4.3, let us consider $U' = \{v_3, v_7\}$ and $U = \{v_3, v_7, v_{11}\}$, we can notice that $max_S(U') = max_S(U)$ and $\mathcal{N}(U') = \mathcal{N}(U)$. This means that all the descriptions $X \subseteq \mathcal{N}(U')$ also cover U. In this example we have $clo(U') = U$.

We aim to only enumerate closed patterns $(clo(U), max_S(U))$, to this end, SIAS-Miner uses the divide and conquer algorithm designed to efficiently compute closed structures described in [36]. Initially, SIAS-Miner is called with $U = \emptyset$ and $C = V$. In each recursive call, SIAS-Miner chooses a candidate $v \in C$ considering two optimizations to obtain a more balanced enumeration tree. If $U = \emptyset$, a vertex v is selected according to its degeneracy order [74] to prioritize first the vertices that should lead to small graphs. If $U \neq \emptyset$, a vertex v is selected following the fail first principle, i.e. the vertex that leads to the smallest characteristic so that to backtrack as soon as possible. Then, the closure of $clo(U \cup \{v\})$ is computed and stored in U'. If U' is included in $U \cup C$, we have the guarantee that U' has not been enumerated yet and the enumeration process continues. The candidates that cannot be added anymore

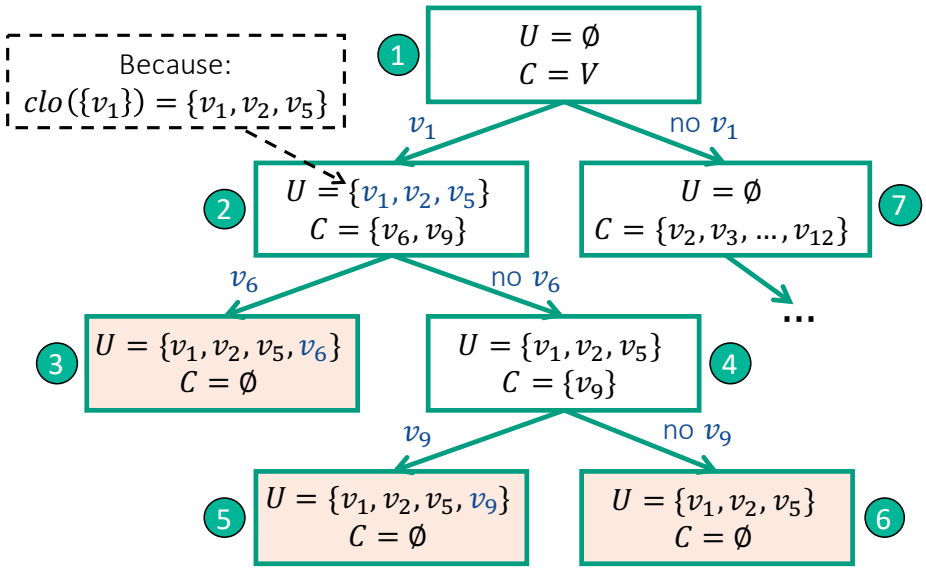


Figure 4.6: Application of SIAS-Miner to the toy example given in Fig. 4.5 with $D_N = 1$. The first enumeration steps.

to U' are pruned and SIAS-Miner is recursively called on U' (Line 8). Another recursive call is also made to enumerate subgraphs that do not contain $\{v\}$ (Line 9). When C is empty, the closed pattern $(U, max_S(U))$ is stored in $Result$, and DL$_V$-Optimise is called to compute DL$_V(U)$ that is finally stored in $minDesc$.

In Fig. 4.6, we show the first enumeration steps of SIAS-Miner when applied to the toy graph of Fig. 4.5 with $D = 1$. SIAS-Miner starts from the root ($U = \emptyset$ and $C = V$), then goes to Step 2, then Step 3... Let us explain how SIAS-Miner generates Step 2 from Step 1. SIAS-Miner chooses v_1 as the next candidate. This initially gives $U = \{v_1\}$, $\mathcal{N}(U) = \{N_1(v_1), N_1(v_2), N_1(v_5)\}$, and $max_S(U) = \{(college, [0, 0.3]), (professional, [0, 0.3])\}$. Afterthat, U is extended using the closure operator clo to add candidates that consevre the current $max_S(U)$ and $\mathcal{N}(U)$. It results that $U = \{v_1, v_2, v_5\}$. Also, the set of remaining candidates becomes $C = \{v_6, v_9\}$, because the other candidates do not share a neighborhood and a characteristic with U. Then, SIAS-Miner continues to Step 3. When $C = \emptyset$ (like in Step 3), the pattern $(U, max_S(U))$ is added to the result set, and SIAS-Miner comebacks to the previous recursive call to continue the rest of the enumeration.

4.5.2 Computing DL$_V(U)$

Computing DL$_V(U)$ is NP-Hard. In fact, if we replace $(log(|\cap_{x \in X} x|)$ in desc(X, U) with a constant value, this problem becomes equivalent to the weighted set cover: it consists in finding the optimal cover of the set $\overline{U}$ based on unions of complements $\overline{N_i(v)}$ and exceptions $\{v\}$ such that $v \in \overline{U}$. Nevertheless, we propose a branch-and-bound approach that takes benefit from several optimisation techniques to solve

Algorithm 4: DL$_V$-Optimise(U, X, Cand, bestDesc)

Input: U the set to describe, X, $Cand \subseteq \mathcal{N}(U)$ the current description and the candidates, bestDesc the current best description found.

Output: bestDesc the best description of the current search sub-space.

```
1 if LB(X, U, Cand) < f(bestDesc, U) then
2     if |X| < α and Cand ≠ ∅ then
3         pruneUseless(U, X, Cand)
4         e ← argmin_{e'∈Cand} f(X ∪ {e'}, U)
5         DL_V-Optimise(U, X ∪ {e}, Cand \ {e}, bestDesc)
6         DL_V-Optimise(U, X, Cand \ {e}, bestDesc)
7     else if f(X, U) < f(bestDesc, U) then
8         bestDesc ← X
```

Algorithm 5: pruneUseless(U, X, Cand)

```
1 Cand ← {e ∈ Cand | gain({e}, X, U) > 0}
```

this problem on instances of interest.

In order to find the optimal description of a set of vertices U, we explore the search space $2^{\mathcal{N}(U)}$ with a branch-and-bound approach described in Algorithm 4. Let X and Cand be subsets of $\mathcal{N}(U)$ that are respectively the current enumerated description and the potential candidates that can be used to describe U. Initially, DL$_V$-Optimise is called with $X = \emptyset$ and Cand $= \mathcal{N}(U)$. In each call, a neighborhood $e \in$ Cand is chosen and used to recursively explore two branches: one made of the descriptions that contain e (by adding e to X), and the other one made of descriptions that do not contain e (by removing e from Cand). Several pruning techniques are used in order to reduce the search space and are detailed below.

Function LB (line 1) lower bounds the lengths of the descriptions that can be generated in the subsequent recursive calls of DL$_V$-Optimise. If LB is higher or equal than the length of the current best description of U desc(bestDesc, U), there is no need to carry on the exploration of the search subspace as no further description can improve DL$_V(U)$. The principle of LB is to evaluate the maximum reduction in exceptions that can be obtained when description X is extended with neighborhoods of Y: $gain_Y(X, U) = |\text{exc}(X, U)| - |\text{exc}(X \cup Y, U)|$, with $Y \subseteq$ Cand. This function can be rewritten using neighborhood complements as $gain_Y(X, U) = |\cup_{y \in Y} (\overline{y} \cap \text{exc}(X, U))|$[4]. We obtain then a simple upper bound of the gain function using the ordered set $\{g_1, ..., g_{|\text{Cand}|}\}$ of $\{gain_{\{e\}}(X, U) \mid e \in \text{Cand}\}$ such that $g_i \geq g_j$ if $i \leq j$:

Property 4.1. $gain_Y(X, U) \leq \sum_{i=1}^{|Y|} g_i$, for $Y \subseteq$ Cand.

[4] $= |\text{exc}(X, U)| - |\text{exc}(X \cup Y, U)| = |(\cap_{x \in X} x) \setminus U| - |(\cap_{e \in X \cup Y} e) \setminus U|$,
$= |(\cap_{x \in X} x) \cap \overline{U}| - |((\cap_{x \in X} x) \cap \overline{U}) \cap (\cap_{y \in Y} y)|$,
$= |(\cap_{x \in X} x) \cap \overline{U}) \setminus (\cap_{y \in Y} y)| = |(\cap_{x \in X} x) \setminus U) \cap \overline{(\cap_{y \in Y} y)}|$,
$= |\text{exc}(X, U) \cap (\cup_{y \in Y} \overline{y})| = |\cup_{y \in Y} (\overline{y} \cap \text{exc}(X, U))|$.

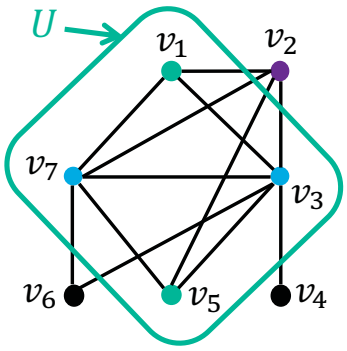


Figure 4.7: A toy example, $U = \{v_1, v_3, v_5, v_7\}$ and $\mathcal{N}(U) = \{N_1(v_2), N_1(v_3), N_1(v_7)\}$, for $D_N = 1$.

Proof. Since the size of the union of sets is lower than the sum of the set sizes, we have $gain_Y(X,U) \leq \sum_{y\in Y} |\overline{y} \cap \text{exc}(X,U)| \leq \sum_{y\in Y} gain_{\{y\}}(X,U) \leq \sum_{i=1}^{|Y|} g_i$.

For instance, in Fig 4.7, we need to describe $U = \{v_1, v_3, v_5, v_7\}$ by an intersection of elements from $\mathcal{N}(U) = \{N_1(v_2), N_1(v_3), N_1(v_7)\}$, given that the maximum range of distance for descriptors is $D = 1$. Let us consider that at some moment of the exploration we have the description $X = \{N_1(v_3)\}$, and the remaining candidates $\text{Cand} = \{N_1(v_2), N_1(v_7)\}$. At this moment, the exceptions are: $\text{exc}(X,U) \triangleq \{v_2, v_4, v_6\}$. In order to compute the upper bound of $gain_Y(X,U)$ for $Y \subseteq \text{Cand}$, we need to compute the ordered set of $gain_{\{e\}}(X,U)$ for $e \in \text{Cand}$. We have $gain_{\{N_1(v_2)\}}(X,U) = 2$ because adding $N_1(v_2)$ allows to remove the exceptions $\{v_4, v_6\}$, and $gain_{\{N_1(v_7)\}}(X,U) = 1$ because adding $N_1(v_7)$ removes only one exception v_4. Thus, the ordered list of gains is $\{2, 1\}$. This means that, for $Y \subseteq \text{Cand}$, if $|Y| = 1$ then $gain_Y(X,U) \leq 2$, and if $|Y| = 2$, then $gain_Y(X,U) \leq 3$. This is the foundation of the function LB defined in what follows.

Definition 4.8. Given the current enumerated description $X \subseteq \mathcal{N}(U)$, the set of vertices U to describe, and the set of potential candidates $\text{Cand} \subseteq \mathcal{N}(U) \setminus X$, the lower bound function LB is defined as:

$$LB(X,U,\text{Cand}) = \min_{i\in[\![0,|\text{Cand}|]\!]} \{(|X| + i + 1) \times \log(|\mathcal{N}|) + \left(1 + \max\left(|\text{exc}(\mathcal{N}(U),U)|, |\text{exc}(X,U)| - \sum_{j=1}^{i} g_j\right)\right) \times \log\left(|U| + \max\left(|\text{exc}(\mathcal{N}(U),U)|, |\text{exc}(X,U)| - \sum_{j=1}^{i} g_j\right)\right)\}.$$

Property 4.2. $\text{desc}(X \cup Y, U) \geq LB(X,U,\text{Cand}), \forall Y \subseteq \text{Cand}$.

Proof. Considering that $|\text{exc}(X \cup Y, U)| = |\text{exc}(X,U)| - gain_Y(X,U)$, and using Property 4.1: we have $|\text{exc}(X \cup Y, U)| \geq |\text{exc}(X,U)| - \sum_{j=1}^{|Y|} g_j$. Also, since the number of exceptions is reduced

when X is extended, the minimum number of exceptions that we can reach in all the search space is $|\text{exc}(\mathcal{N}(U), U)|$. Thus, we have the following lower bound for $|\text{exc}(X \cup Y, U)|$:

$$|\text{exc}(X \cup Y, U)| \geq \max(|\text{exc}(\mathcal{N}(U), U)|, |\text{exc}(X, U)| - \sum_{j=1}^{|Y|} g_j) \tag{4.4}$$

$\text{desc}(X \cup Y, U)$ is monotonically increasing w.r.t. $\text{exc}(X \cup Y, U)|$. If we replace $\text{exc}(X \cup Y, U)|$ by its lower bound from Equation 4.4 in $\text{desc}(X \cup Y, U)$, this will give a lower bound for desc:

$$\begin{aligned}
&\text{desc}(X \cup Y, U) \geq (|X| + |Y| + 1) \times \log(|\mathcal{N}|) + \\
&\left(1 + \max\{|\text{exc}(\mathcal{N}(U), U)|, |\text{exc}(X, U)| - \sum_{j=1}^{|Y|} g_j\}\right) \\
&\cdot \log(|U| + \max\{|\text{exc}(\mathcal{N}(U), U)|, |\text{exc}(X, U)| - \sum_{j=1}^{|Y|} g_j\}),
\end{aligned}$$

But also, by definition: $LB(X, U, \text{Cand}) \leq (|X| + |Y| + 1) \times \log(|\mathcal{N}|) + (1 + \max\{0, |\text{exc}(X, U)| - \sum_{j=1}^{|Y|} g_i\}) \cdot \log(|U| + \max\{|\text{exc}(\mathcal{N}(U), U)|, |\text{exc}(X, U)| - \sum_{j=1}^{|Y|} g_i\})$. So, by transitivity $LB(X, U, \text{Cand}) \leq f(X \cup Y, U)$, and this concludes the proof. ∎

In other terms, in the recursive calls, a description length will never be lower than $LB(X, U, \text{Cand})$. Thus, if its value is higher than the current best description, it is certain that no better description can be found.

Function pruneUseless (line 3) removes candidate elements that can not improve the description length, that is candidates $e \in \text{Cand}$ for which $gain(\{e\}, X, U) = 0$. Such element does not have the ability to reduce the number of exceptions in X. This also implies that e will not reduce the number of exceptions for descriptions $X \cup Y$, with $Y \subseteq \text{Cand}$. Thus, such elements will not decrease the description length of $X \cup Y$.

In Fig. 4.7, for $X = \{N_1(v_7)\}$ and $\text{Cand} = \{N_1(v_2), N_1(v_3)\}$, the exceptions are: $\text{exc}(X, U) = \{v_2, v_6\}$. All these exceptions also belong to $N_1(v_3)$ which is a candidate. For this part of the search space, $N_1(v_3)$ is not able to reduce the number of exceptions, so it is not able to improve (reduce) the value of desc. pruneUseless allows to prune this sort of candidates.

The last optimisation consists in choosing $e \in \text{Cand}$ that minimises $\text{desc}(X \cup \{e\}, U)$ (line 5 of Algorithm 4). This makes it possible to quickly reach descriptions with low DL_V, and subsequently provide effective pruning when used in combination with LB.

4.5.3 Time complexity of SIAS-Miner

Theoretically, the number of closed CSEA patterns can be exponential in the size of the input dataset. Since SIAS-Miner computes all the closed patterns, the number of enumeration steps can be exponential too. Still, we can study the delay complexity (worst case complexity between two enumeration steps).

By denoting by $\mathcal{C}_{DL_V}$ the complexity of DL_V-Optimise, the delay of SIAS-Miner is in $\mathcal{O}(\max\{|V|^2 \cdot (|V|+|A|), \mathcal{C}_{DL_V}\})$. In fact, the complexity of making one recursive call of SIAS-Miner when $C \neq \emptyset$ is $\mathcal{O}(|V| \cdot (|V|+|A|)|)$, which corresponds to the cost of Line 6 (the other lines have lower complexities). SIAS-Miner enumerates the closed patterns in a depth-first manner. The depth of the search space is bounded by $|V|$, as in each recursive call at least one element $v \in V$ is removed from C. Thus, the number of recursive calls between two leaves is bounded by $2 \cdot |V|$. Between two enumerated patterns, DL_V-Optimise is called once. Therefore, the time delay of SIAS-Miner is $\mathcal{O}(\max\{|V|^2 \cdot (|V|+|A|), \mathcal{C}_{DL_V}\})$.

It remains to compute $\mathcal{C}_{DL_V}$ the complexity of DL_V-Optimise. In the general case, DL_V-Optimise can make at worst $2^{|V|}$ recursive calls to compute $\text{DL}_V(U)$. However, in concrete applications, the parameter α, that controls the size of the description, is set to a small value (e.g., generally $\alpha = 2$). Indeed, in our experiments, the majority of top patterns are described with at most two neighborhood intersections. For $\alpha = 2$, there is at most $\frac{|V|(|V|+1)}{2}$ possible descriptions, and the cost of each recusrive call of DL_V-Optimise is $\mathcal{O}(|V|^2)$. Then, it gives a total complexity of $\mathcal{O}(|V|^4)$ to compute $\text{DL}_V(U)$ if $\alpha = 2$. In a more general manner, the complexity of computing $\text{DL}_V(U)$ is at most $\mathcal{O}(|V|^{2+\alpha})$.

Thus, for a specific value of α, SIAS-Miner has a polynomial time delay, and the worst case complexity of this delay is at most $\mathcal{O}(\max\{|V|^2 \cdot (|V|+|A|), |V|^{2+\alpha}\})$. We show in Section 4.6 that SIAS-Miner is able to run on real world datasets with tens of thousands vertices, even if α is set to its maximum value $|V|$.

4.5.4 Using SIAS-Miner to iteratively mine CSEA patterns

The output of SIAS-Miner consists of the list of all the patterns sorted by their initial SI measure. In order to iteratively mine CSEA patterns with the application of the model updating proposed in Section 4.4, we can run SIAS-Miner once and apply a lazy sorter on its output. At the beginning of each iteration, this sorter picks the pattern with the highest SI, and proceeds to the update of SI of the other patterns in the previously sorted list $L = \{P_0, ..., P_q\}$. However, we do not need to update the SI of all the patterns in L: we stop as soon as we find a pattern P_i whose updated SI is the highest among the already updated patterns $\{P_0, ..., P_{i-1}\}$, but also higher than the SI of the patterns that are not updated yet: $\{P_{i+1}, ..., P_q\}$. In fact, the SI of all these remaining patterns will either decrease or remains unchanged, so P_i will necessarily be the best pattern of the next iteration. This procedure can be then repeated iteratively to get more patterns.

4.6 Experiments

In this section, we report our experimental results. We start by describing the real-world datasets we used, as well as the questions we aim to answer. Then, we provide a thorough comparison with state-of-the-art algorithms: CEnergetics [24] (presented in Chapter 3), P-N-RMiner [141], and GAMER [98]. Eventually, we provide a qualitative analysis that demonstrates the ability of our approach to achieve the

desired goal. For reproducibility purposes, the source code and the data are made available[5].

Experimental setting. Experiments are performed on four datasets:

− The `London` graph ($|V| = 289, |E| = 544, |\hat{A}| = 10$) is based on the social network Foursquare[6]. Each vertex depicts a district in London and edges link adjacent districts. Each attribute stands for the number of places of a given type (e.g. outdoors, colleges, restaurants, etc.) in each district. The total number of represented venues in this graph is 25029.

− The `Ingredients` graph ($|V| = 2400, |E| = 7932, |\hat{A}| = 20$) is built from the data provided by Kaggle[7] and features given by Yummly[8]. Each vertex is a recipe ingredient. The attributes correspond to the number of recipes grouped by nationality (greek, italian, mexican, thai, etc.) that include this ingredient. An edge exists between two ingredients if the Jaccard similarity between their recipes is higher than 0.03. The total number of used recipes is 39774.

− The US `Flights` graph ($|V| = 322, |E| = 2039, |\hat{A}| = 14$) is a dataset provided by Kaggle[9] which contains information about flights between U.S.A airports in 2015. The vertices represent U.S.A airports and the attributes depict the number of flights per airline company in the corresponding airports. Edges connect two airports if there are at least 100 flights between them.

− The `DBLP` graph ($|V| = 38,895, |E| = 112,404, |\hat{A}| = 10$) is a co-authorship graph built from the DBLP digital library. Each vertex represents an author who has published at least one paper in 10 major conferences and journals of in the Data Mining and Databases communities[10], between January 1990 and March 2019. Each edge links two authors who co-authored at least one paper in these conferences and journals. The attributes depict for each author the number of publications in the 10 aforementioned conferences and journals.

Considering all the numerical values of attributes is computationally expensive, we pre-process each graph so that for each attribute, the values $c_a(v)$ are binned into five quantiles. This means that each attribute $c_a(v)$ is numerical and takes only 5 different values.

There is no approach that supports the discovery of subjectively interesting attributed subgraphs in the literature. Nevertheless, we identify some approaches whose goal share some similarities with ours. We consider them in our study:

− P-N-RMiner [141] is an algorithm that mines multi-relational datasets to enumerate a specific structure of patterns called Maximal Complete Connected Subsets (MCCS). Any vertex-attributed graph can be mapped to an entity-relational model where the pattern syntax of P-N-RMiner is equivalent to ours (each MCCS corresponds to a closed pattern $(clo(U), max_S(U))$ in our context). This means that P-N-RMiner is very suitable to evaluate the performance of our algorithm. Although the pattern syntax in this design is equivalent to our approach, our interestingness quantification is very different, because the information contained in the patterns shown to the user does not align with the ranking of P-N-RMiner. Hence, we

[5] `http://goo.gl/ZxsvbX`
[6] https://foursquare.com
[7] `https://www.kaggle.com/kaggle/recipe-ingredients-dataset`
[8] `https://www.yummly.com/`
[9] https://www.kaggle.com/usdot/flight-delays/data
[10] DMKD, VLDB journal, Machine Learning, TKDE, KDD, ICDM, VLDB, ICDE, ICML, IJCAI

use P-N-RMiner only to compare the runtime performance of our approach.
− CEnergetics [24] (presented in Chapter 3) aims at discovering connected subgraphs involving overrepresented and/or underrepresented attributes. It assesses exceptionality with the weighted relative accuracy (WRAcc) measure that accounts for margins but cannot account for other background knowledge.
− GAMER [98]: Given an attributed graph, this method finds dense subgraphs (quasi-cliques) where vertices show a high similarity in subsets of attributes. In these subgraphs, these attribute values fall into narrow intervals whose width does not exceed a specified threshold W. The main difference with our approach is that GAMER looks only for similarity and cohesiveness, but not exceptionality and surprisingness.

In this experimental study, our aim is to answer the following questions: What is the efficiency of SIAS-Miner regarding to graph dimensions? Is SIAS-Miner able to deal with real world datasets? What are the differences between the results of our approach and those of the considered baselines? What about the relevance of the CSEA patterns?

Quantitative experiments. Fig. 4.8 reports the runtime of SIAS-Miner and P-N-RMiner according to the number of vertices, the number of attributes and the minimum number of vertices of searched patterns, for each of the datasets. The points that are not displayed in the curves of P-N-RMiner are the ones that exceeded a time limit of 10^4 seconds. For example, when we variated the attributes in the ingredients dataset, P-N-RMiner was not able to finish any configuration in less than 10^4 seconds. These tests reveal that SIAS-Miner outperforms P-N-RMiner in all the datasets in almost all the configurations, and the difference is generally between 2 and 4 orders of magnitude. Although P-N-RMiner is a principled algorithm that uses several advanced optimization techniques, SIAS-Miner is faster since it is particularly defined to deal with attributed graphs, and it takes benefits from several specificities of these structures to optimize the search space exploration.

Qualitative experiments. The goal is to compare the properties of the patterns found by SIAS-Miner with those of CEnergetics and GAMER. We do not consider P-N-RMiner for two reasons: (1) Even if the pattern syntax can be made equivalent between SIAS-Miner and P-N-RMiner, the model used in P-N-RMiner to assess the quality of patterns is not adapted to the goal of mining attributed graphs, (2) P-N-RMiner was not able to finish it execution in any of the studied datasets when the whole set of vertices are considered. We first extract the top 200 diversified patterns of each approach. Also, DBLP results for GAMER are not studied because this approach was not able to perform on this dataset. We compute a summary of patterns obtained by CEnergetics based on Jaccard similarity in order to have diversified patterns to study. This set only contains patterns whose pairwise Jaccard similarity is lower than 0.6. This step is not applied on GAMER and SIAS-Miner results since GAMER output is internally summarized with a similar approach, and SIAS-Miner patterns are already diversified thanks to the iterative mining with model updating. In the following, we compare the properties depicted in Fig 4.9:
− Density and relative degree: The density of a pattern (U, S) is $\frac{2\times|E(U)|}{|U|\times(|U|-1)}$ where $|E(U)|$ is the number of edges in the induced subgraph $G[U]$, and the relative degree of a vertex v in a set of vertices U is $\frac{|N_1(v)\cap U|}{|U|-1}$. These properties are the highest for GAMER, this can be explained by the fact that its

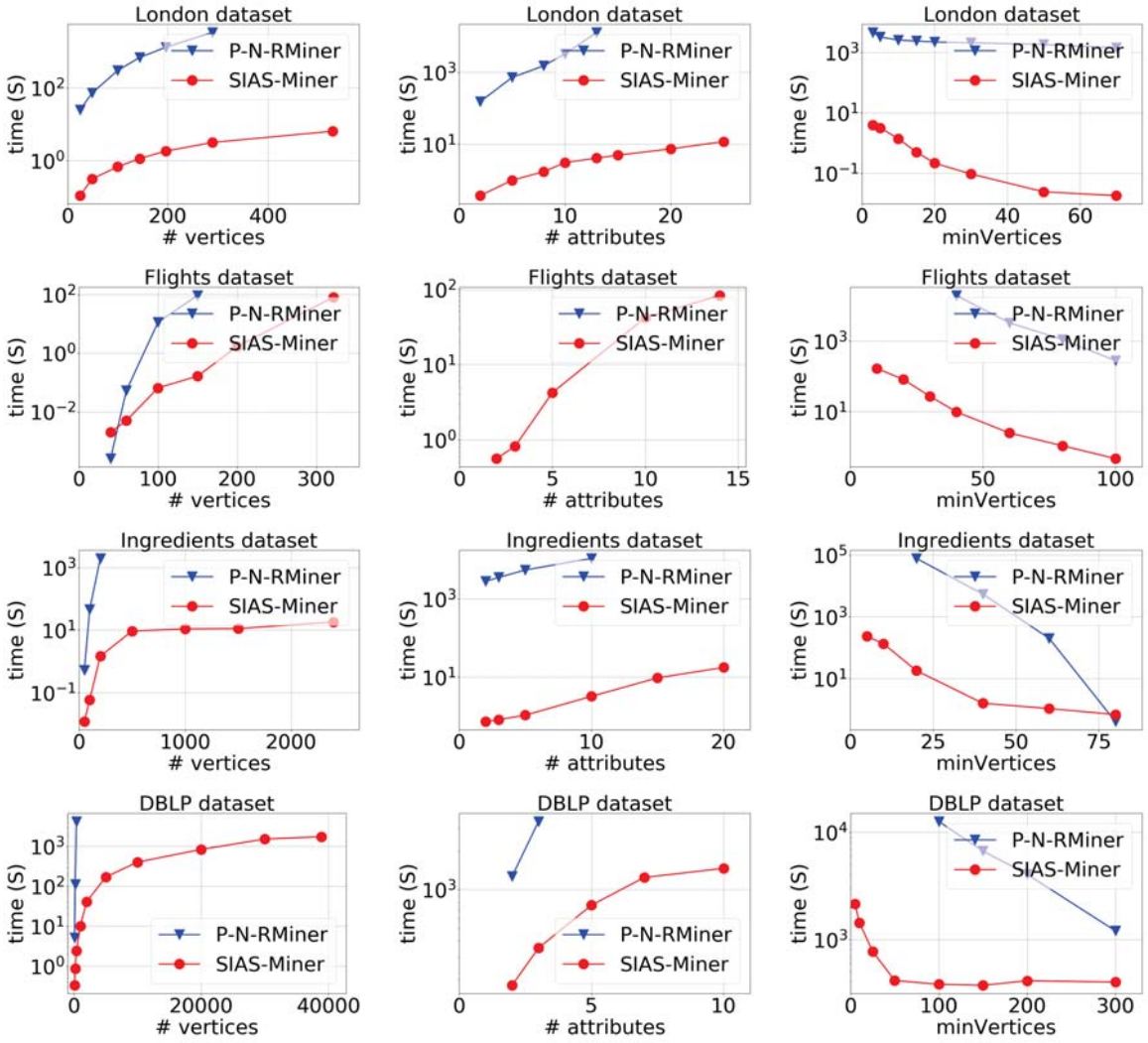


Figure 4.8: SIAS-Miner vs P-N-RMiner: runtime when varying $|V|$ (1st column), $|A|$ (2nd column) and a threshold on the minimum number of vertices in searched patterns (3rd column) for London graph ($D_N = 3$, 1st row), flights graph ($D_N = 1$, 2nd row), ingredients graph ($D_N = 1$, 3rd row), and DBLP dataset ($D_N = 1$,4th row).

patterns are made of quasi-cliques, which makes them denser. In `Flights` and `Ingredients` datasets, the density and the degree for SIAS-Miner are higher than those of CENERGETICS, while they are lower for the `London` graph. In fact, for SIAS-Miner, D_N was set to 3 for `London`, and to 1 for `Flights` and `Ingredients`. The higher the value of D_N, the sparser the results can be. In general, CENERGETICS patterns have a low density and relative degree, because this approach requires only the connectivity of vertices in the patterns, some of these patterns can be very sparse subgraphs.

− Diameter: the diameter of a subgraph U is the maximum pairwise distance between vertices of U. GAMER patterns have the smallest average diameter, while CENERGETICS patterns have the highest ones. The diameter of SIAS-Miner is comparable to the one of CENERGETICS in `London` ($D_N = 3$) and `DBLP` ($D_N = 1$), but it is smaller for `Flights` and `Ingredients` ($D_N = 1$).

− Size and number of covered attributes: The size of a pattern (U, S) corresponds to $|U|$ and the number of covered attributes is $|S|$. GAMER has patterns with the smallest average size, this is reasonable because it requires a harder constraints on the structure of patterns, which is the quasi-cliqueness. This small size of U allows GAMER patterns to be covered with a larger number of attributes comparing with the other approaches.

− Contrast of attributes: given the identified patterns (U, S) we want to measure how much the characteristics S are over (or under) expressed in U. First, we define the constrast of a given attribute a in a given set of vertices U as the absolute difference between its average ratio in U and its overal average ratio: $contrast(a, U) = |\frac{1}{|U|} \times \sum_{v \in U} \frac{\hat{a}(v)}{\hat{A}(v)} - \frac{1}{|V|} \times \sum_{v \in V} \frac{\hat{a}(v)}{\hat{A}(v)}|$, with $\hat{A}(v) = \sum_{a \in A} \hat{a}(v)$. The contrast of a pattern (U, S) is the average contrast $contrast(a, U)$ among the attributes a that appear in S. As expected, GAMER has the minimum values of contrasts for all the datasets, indeed, GAMER is only interested by the similarity of attributes in the pattern but not by their exceptionality. The contrasts for SIAS-Miner and CENERGETICS are higher, and they are comparable in `London Flights` and `DBLP` datasets, while it is higher for SIAS-Miner in `Ingredients` dataset.

To conclude, there is a clear structural difference between patterns of the three approaches. GAMER finds denser subgraphs, and CENERGETICS patterns are generally the sparsest ones. GAMER does not look to the exceptionality of attributes in the patterns, but only for their similarities. Another major difference is the possibility of integrating different prior beliefs in the MaxEnt model of SIAS-Miner, and the update of the background model after the observation of each pattern by the user, which is not possible in the other attributed graph mining approaches.

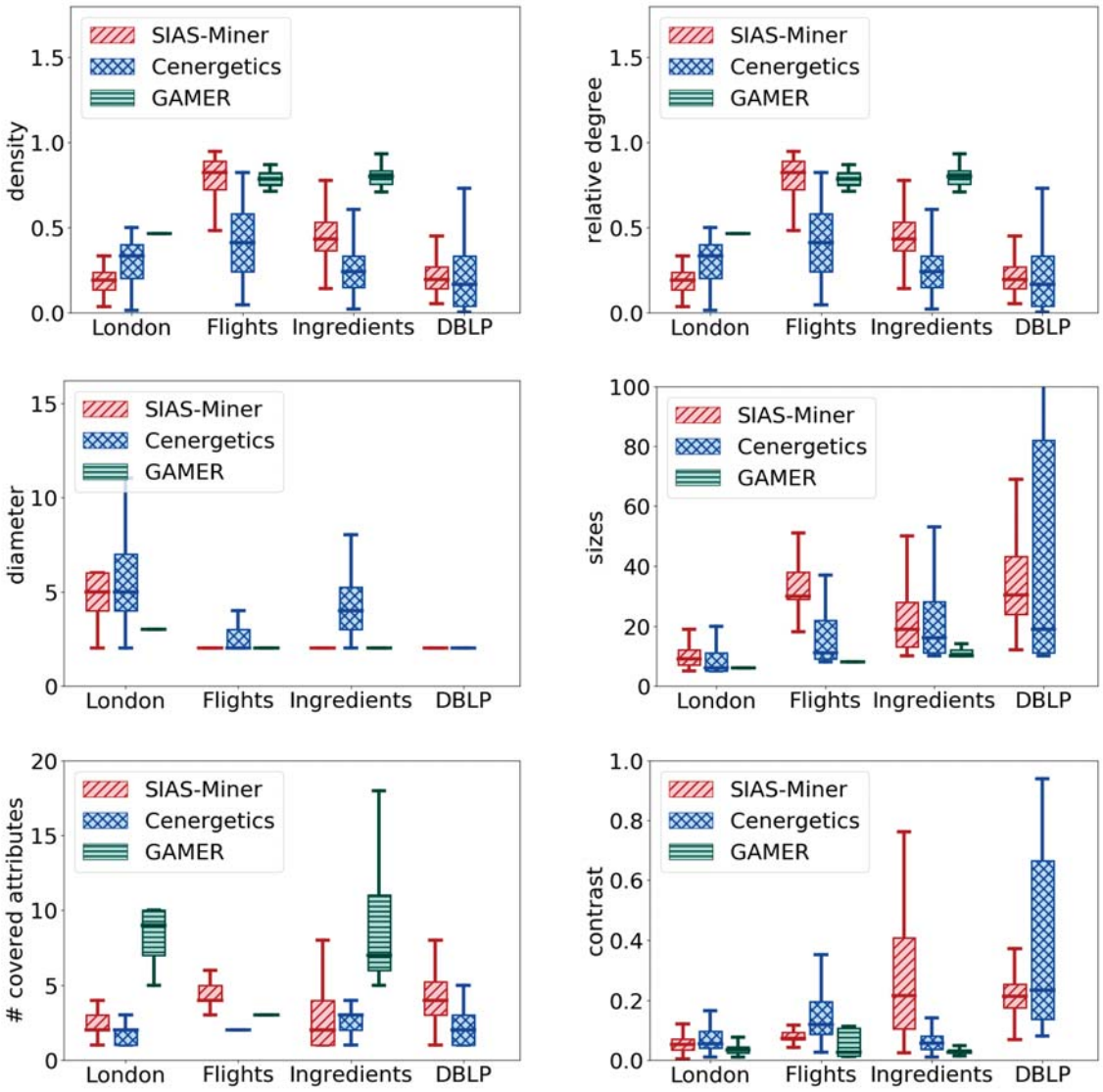
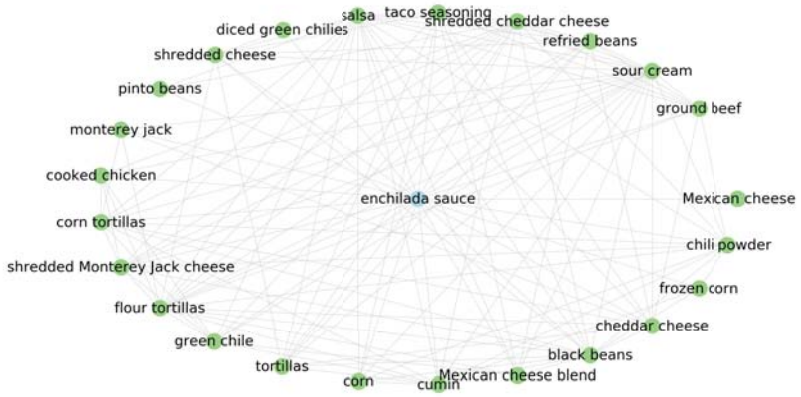


Figure 4.9: Comparisons of properties of identified patterns by SIAS-Miner, CENERGETICS, and GAMER in `London`, `Ingredients`, and `Flights` graphs.

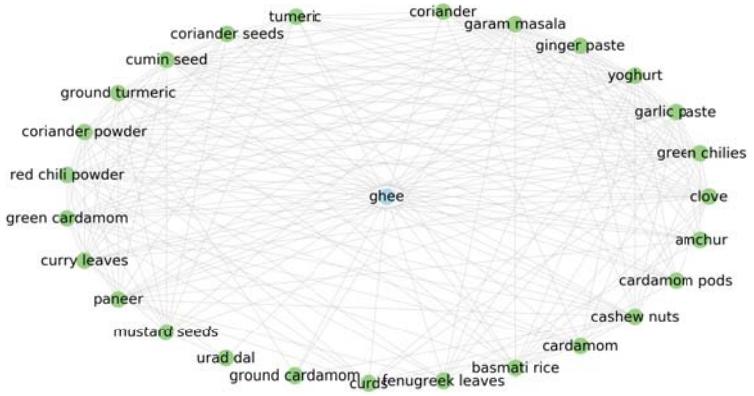
Illustrative results. In Fig. 4.1, we show some patterns that SIAS-Miner discovered in the `London` graph. We chose the top 4 patterns, and P_{14} which is the best pattern with more than one core vertex. Green cells represents vertices covered by a CSEA pattern while blue cells are the core vertices that are also in the cover, and the red cells are normal exceptions. For example, P_1 covers the distance three neighbors of the blue vertex, with an exceptional prevalence of nightlife and professional venues. Indeed, P_1 includes the City of London which contains the primary central business district (CBD) of London, and it is known by its prevalent number of nightlife venues (pubs, bars,etc.) in some areas like the Soho and the South Bank neighborhood.

We report in Fig. 4.10 the top 4 patterns discovered by SIAS-Miner in `Ingredients` graph. P_1 corresponds to a set of ingredients that appear a lot in Mexican recipes. They are described as neighbors of enchilada sauce which is an ingredient originally from Mexico, notice that this patterns does not contain any exception. In Figure 4.12, we also present some other specific patterns, P_{15} is the best pattern with more than one attribute in the characteristic (an exceptional prevalence in Chinese, Thai and Japanese food), and P_{71} the best pattern with more than one core vertex (the cores are garlic cloves and chili powder). This last pattern has also an over expression in several types of cuisine, but the prevalence is much more significant in Mexican food (with $C_{mexican}(v) \leq 8.6 \cdot 10^{-4}$). For more details, we display in the companion page the complete set of top 100 patterns of each of `London` and `Ingredients` datasets.

We show in Figure 4.11 the top 3 patterns discovered by SIAS-Miner in `DBLP` graph. P_1 consists in researchers who are co-authors of Milind Tambe in some of the 10 studied conferences and journals. They have published a significant number of papers in IJCAI conference, and they have never published in VLDB and VLDB Journal. All the co-authors of Yoshio Bengio are included in P_2, except 5 of them. They publish a lot in ICML and none of them have published in VLDB Journal or DMKD. We Also show in Figure 4.13 the best `DBLP` pattern with more than one core vertex. It corresponds to the common co-authors of Hui Xiong and Enhong Chen, for whom there is a large number of publications in ICDM, a low number of ICML papers, and no publication in VLDB.

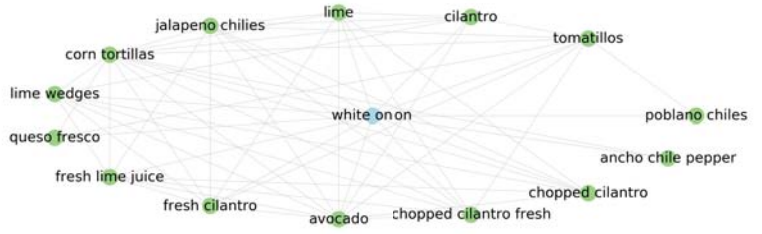


P_1 : {Mexican:$[0, 10^{-18}]$}$^+$, $SI(P_1) = 40.34$, $|U| = 25$, exceptions number: 0



P_2 : {Indian:$[0, 10^{-18}]$}$^+$, $SI(P_2) = 27.03$, $|U| = 27$, exceptions number: 5

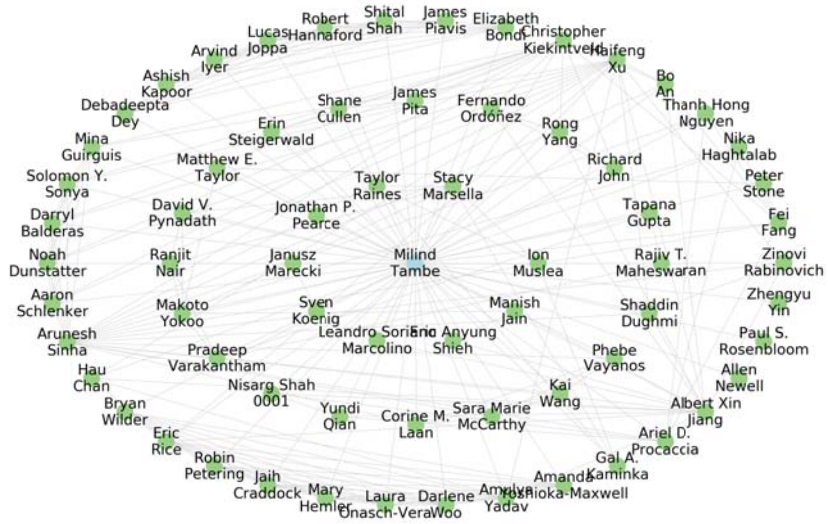
P_3 : {Italian:$[0, 10^{-18}]$}$^+$, $SI(P_3) = 24.23$, $|U| = 18$, exceptions number: 2



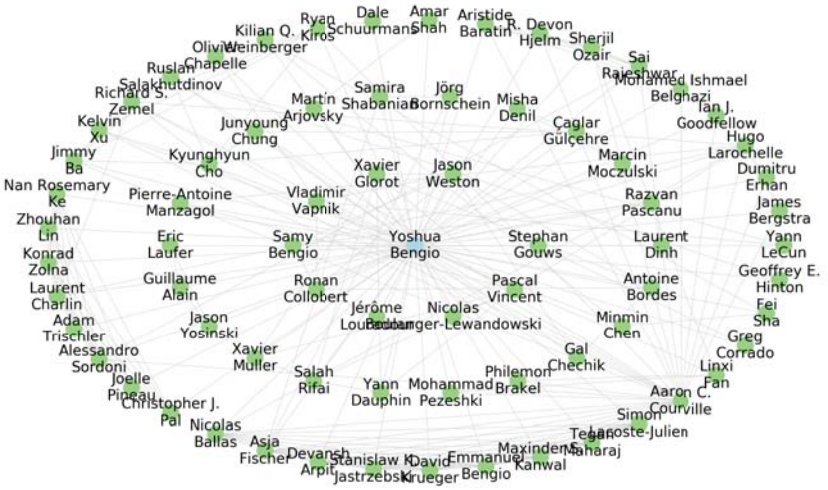
,

P_4 : {Mexican:$[0, 10^{-18}]$}$^+$, $SI(P_4) = 17.46$, $|U| = 15$, exceptions number: 3

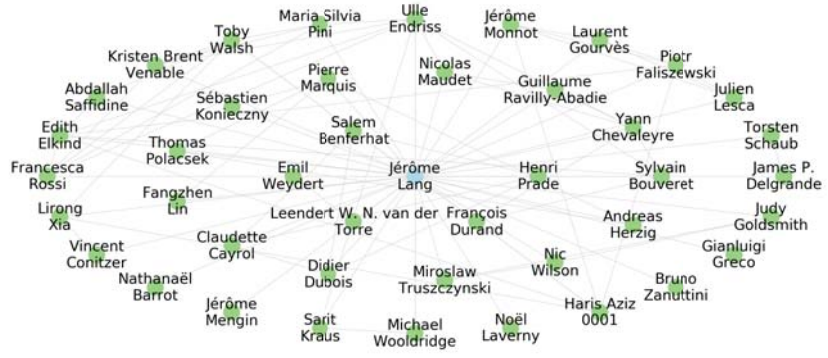
Figure 4.10: Top 4 patterns discovered in `Ingredients` ($minVertices = 5$, $D_N = 1$).



P_1 : {IJCAI:$[0, 0.30]$}$^+$, {VLDB:$[1, 1]$, VLDB Journal:$[1, 1]$}$^-$
$SI(P_1) = 3.47$, $|U| = 68$, exceptions number: 0

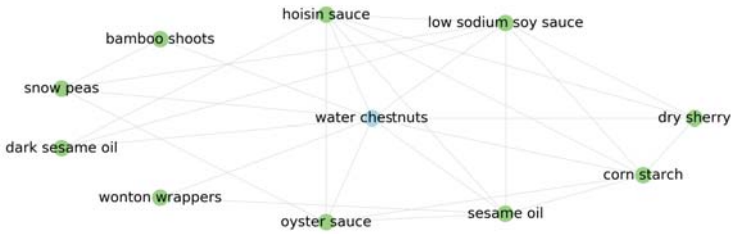


P_2 : {ICML:$[0, 0.15]$}$^+$, {VLDB journal:$[1, 1]$, DMKD:$[1, 1]$}$^-$
$SI(P_2) = 3.22$, $|U| = 73$, exceptions number: 5

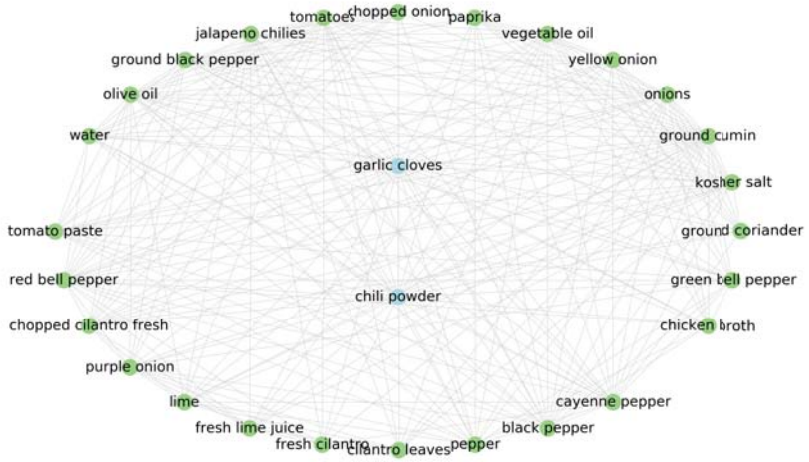


P_3 : {IJCAI:$[0, 0.15]$}$^+$, {KDD:$[1, 1]$, VLDB:$[1, 1]$, VLDB Journal:$[1, 1]$}$^-$
$SI(P_3) = 3.18$, $|U| = 43$, exceptions number: 0

Figure 4.11: Top 3 discovered in the DBLP graph. P_{15} ($minVertices = 5$, $D_N = 1$).

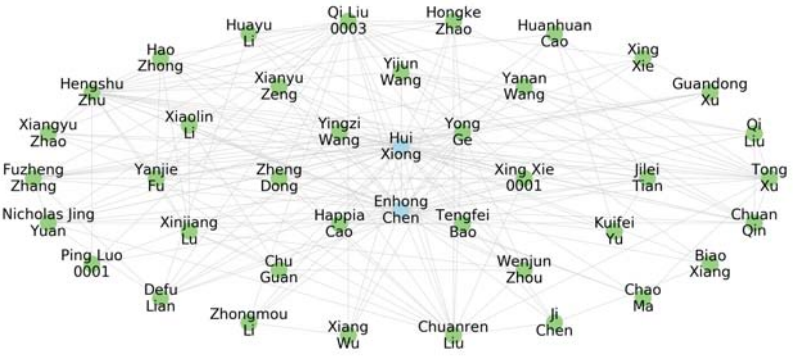


P_{15} : {Chinese:$[0, 10^{-18}]$, Thai:$[0, 0.35]$, Japanese:$[0, 0.44]$}$^+$
$SI(P_{15}) = 6.58$, $|U| = 11$, exceptions number: 5



P_{71} : {Mexican:$[0, 8.6 \cdot 10^{-4}]$, Indian:$[0, 0.22]$, Thai:$[0, 0.35]$, Spanish:$[0, 0.49]$}$^+$
$SI(P_{71}) = 1.21$, $|U| = 28$, exceptions number: 5

Figure 4.12: Patterns discovered in the `Ingredients` graph. P_{15}: the best pattern with more than one attribute in the characteristic, P_{71}: the best pattern with more than one core vertex in the description. ($minVertices = 5$, $D_N = 1$).



P_1 : {ICDM:$[0, 0.30]$}$^+$, {ICML:$[0.7, 1]$, VLDB:$[1, 1]$}$^-$
$SI(P_1) = 1.47$, $|U| = 40$, exceptions number: 4

Figure 4.13: DBLP graph: Best pattern with more than one core vertex ($minVertices = 5$, $D_N = 1$).

4.7 Conclusion

This chapter has introduced a novel approach for the problem of mining exceptional attributed subgraphs. We have proposed an improved pattern syntax called *cohesive subgraphs with exceptional attributes* (CSEA), and a new pattern interestingness model. These patterns provide a set of attributes that have exceptional values throughout a cohesive subset of vertices. In contrast to the approach proposed in Chapter 3, the interestingness model defined for CSEA patterns accounts for the background knowledge available about the data, and considers the assimilation cost of patterns. This allows to identify CSEA patterns that are simultaneously informative and easy to assimilate. The interestingness is defined based on information theory, as the ratio of the information content (IC) over the description length (DL). We have also shown how to update the background model when the current knowledge of the user changes by observing new patterns. This makes it possible to iteratively identify patterns that give new information out of the one already aquired by the previously observed patterns, and thus avoid the problem of redundancy in the results set.

In many applications, vertex-attributes are organised as a hierarchy. In the experimental study of this chapter, we have used Foursquare graphs whose vertices are described by types of venues. Foursquare defines a hierarchy among these attributes. For example, the attribute “Restaurant” is a child (a sub-type) of the attribute “Food”. Attributes of different levels of the hierarchy share overlapping information. Telling the user that a subgraph has many “Food” venues would let her increase her expectations about the number of “Restaurant” venues in this subgraph. This kind of dependencies has never been considered when defining a subjective interestingness model. Likewise, the interestingness model of this chapter also ignores these dependencies. In Chapter 6, we address this problem and we extend the subjective interestingness framework to integrate the dependency between hierarchical attributes.

As mentionned in Chapter 2, the notion of incorporating the user in the interestingness model can be interpreted in two different ways: (1) accounting for the background knowledge or (2) integrating the user preferences. The method proposed in the current chapter involves the user from a “background knowledge” point of view. In Chapter 5, the purpose will be to integrate the user-preferences when mining attributed subgraphs, by following the interactive strategy proposed in [73]. The incorporation of the user will be limited to her preferences though, i.e., the background knowledge will be ignored, as it is already a challenging problem to integrate the preferences into the task of mining attributed graphs.

CHAPTER 5

INTEGRATING USER INTEREST IN ATTRIBUTED SUBGRAPH MINING

Contents

5.1 Introduction

In Chapter 4, we proposed an attributed subgraph mining approach that takes into account prior knowledge of the user in the mining process. In this chapter, we aim to study the possibility of considering her preferences when quantifying the quality of patterns. Indeed, a user may be particularly interested in patterns related to some specific topics. We want to propose a user-driven approach where we iteratively update the model so that it better matches with user interests. Several data mining tasks can be addressed with attributed subgraph mining approaches. We focus on the task of user-driven event detection in social media, as we believe that it is a good and concrete case study where the incorporation of user preferences makes sense. Furthermore, this task can target a large number of people who are able to undestand its result and interact with it. This makes it possible to perform experiments with many real users to evaluate to which extent the used model is able to incorporate their preferences when ranking the patterns.

Event detection is one of the most important research topics in social media analysis. Despite this interest, few researchers have addressed the problem of identifying geolocated events in an unsupervised way, and none includes user interests during the process. Section 5.2 provides a detailed review of related works. We tackle the problem of local event detection from social media data. We present a method to automatically identify events by evaluating the burstiness of hashtags in a geographical area and a time interval, and at the same time integrating user feedback. The particularities of this subgraph mining approach comparing with the ones presented in the previous chapters are: (1) Proposition of a new interestigness model specific for the task of event detection, (2) integration of time dimension, i.e., vertex-attributes are functions that are defined for each pair of vertex and timestamp, and, (3) incorporation of user preferences in the interestingness model.

This is the first attempt to the discovery of user-driven events. In our approach, posts (e.g., tweets) are modeled as an attributed graph whose vertices encode geographical areas and edges depict neighborhood relationships. Time series of term occurrences are associated to vertices, i.e., each term can be seen as an attribute that depicts each vertex. In this unsupervised framework, a geolocated event is considered as a set of terms whose number of occurrences is large enough in a connected subgraph and a time interval. Events are then extracted in a time window and the user has the possibility to tag those he/she likes. This interactive process allows to extract events of interest to users. User preferences are then integrated in the quality measure used to define and rank events. This measure conveys user interest and promotes events that involve topics or geographical areas the user is interested in.

Fig. 5.1 describes our approach. From a social network, we extract time-stamped geolocated posts published during a given period. This set of posts, B (e.g., tweets), is used to generate a graph G_H of term co-occurrences. A second graph G_V depicts adjacency relations between geographic areas from which posts were emitted. Those graphs are used to guide the event discovery process making it possible to solve term synonymy problems as well as the spatial variability (size and location). Moreover, these graphs are the support of an interactive process with the user and are used to rank the identified events according to her preferences. By liking or not events, the user selects some events that are then used by the algorithm to derive locations and topics of preference. These preferences are then used to give a

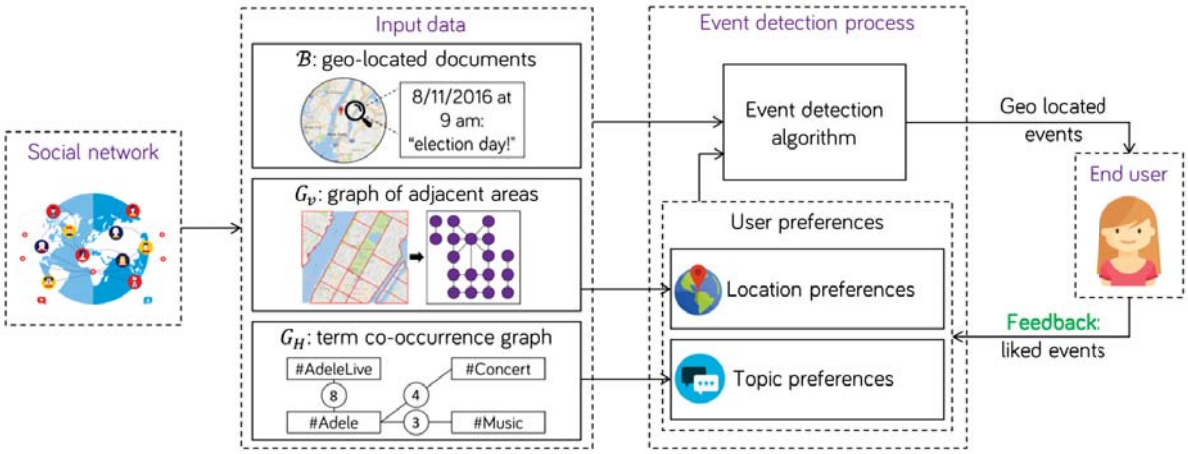


Figure 5.1: Overview of the user-driven event detection system.

greater importance to events related to preferred geographical areas or terms of interest. If there is no interaction with the user, only data-driven events are detected.

The rest of the chapter is organized as follows. Section 5.2 reviews the works related to the task of event detection. Section 5.3 presents a unified framework for data-driven and user-driven geolocated event detection, based on attributed subgraph mining approach. Section 5.4 introduces the user-driven approach. Section 5.5 presents our event detection algorithms: SIGLER-Cov, an efficient event detection algorithm with coverage guarantee, and SIGLER-Samp that directly computes a sampling of the output space of user-diven geolocated events. In Section 5.6, we report a thorough empirical study that provides both quantitative and qualitative performances of our approach. First, we establish that data driven geolocated events cannot be discovered by non location-aware approaches. Second, we compare our method with MED [67] and GeoBurst [227], the state-of-the-art data driven geolocated event mining algorithms. We demonstrate that SIGLER-Cov (1) is several orders of magnitude faster than MED and three times faster than GeoBurst, and (2) is more robust to noise compared to both methods. Finally, the ability of our method to extract interesting user-driven events is experimentally validated. To that end, we ran experiments using a crowdsourcing platform on several cities (e.g., New York, Los Angeles, London) with different settings (e.g., unpaired and paired samples). Section 5.7 concludes and provides future directions.

5.2 Related work on event detection

We focus on space-time event detection from social media with user interaction during the detection process. Detecting events in social media has raised great interest in the last decade. Several papers propose to identify targeted events using supervised [10, 20, 138, 184, 217] or semi-supervised approaches [110, 111]. Such methods require a certain amount of labeled data to detect domain specific events, as opposed to the unsupervised problem considered in this chapter.

In the seminal unsupervised approach [107], events are detected using signal processing methods on word occurrence time series. As several bursty terms can be associated to one real event (synonyms or correlated words), clustering-based approaches [9, 115, 218] are used to group terms with similar temporal patterns. In our experimental study, we show that these approaches fail to detect geo-spatial events. Other methods [5, 53] model the data as a heterogeneous interaction graph with content-related features associated to the nodes. Non-parametric statistics or clustering techniques are then applied on it to extract subgraphs. However, the subgraphs of strong interaction describe hardly local events that are generally not related to direct interactions within the graph. In fact, an event can be characterized with a sparse interaction network corresponding to tweets posted by people who mainly do not know each other. It is important to note that as non location-aware event detection methods fail to detect geolocated events, post-processing such events in order to promote events according to the user interest is not a solution.

Few approaches integrate spatial information during the event detection process. Among them, Triovecevent [228] discovers local events based on multimodal embedding. However, this method is supervised and requires a trained classifier to judge whether a detected cluster of tweets is related to an event or not. Several demo papers [88, 108, 211, 214] present simple systems designed to identify groups of messages that have been posted close in time and space, and assimilate them to events based on the co-occurrence of their terms. A more sophisticated Multiscale Event Detection (MED) method [67] computes the similarity between tweets based on the time series of the occurrences of their common terms considered at different temporal and spatial scales. The most appropriate scale is used to derive a similarity graph on top of which events are detected by a graph-based clustering process. Another state-of-the-art approach is GeoBurst [227]. It is based on an authority measure that captures the geo-topic correlations among tweets. These correlations make it possible to group the tweets using some elaborate clustering techniques. In [227] the authors show that GeoBurst provides better results than two other local event detection approaches: EVENTWEET [2] and WAVELET [54]. EVENTWEET is based on a clustering of spatially and temporally bursting terms. WAVELET uses wavelet analysis to extract events from geo-tagged Flickr photos. Thus, MED and GeoBurst are the main competitive approaches we consider in Section 5.6. We show the superiority of our approach towards these two competitors in terms of computation time and noise robustness.

5.3 A unified framework for data-driven and user-driven geolocated events

In this section, we introduce the problem of data and user driven geolocated event discovery in a unified view. From microblogging social media, we consider a set of posts B, a batch[1], where each element b is described by: (1) the set of terms that appear in b ($b.terms$), (2) the time at which b was sent ($b.time$), (3) the GPS coordinates of the post of b ($b.loc$). From such a batch, we collect the following data: (1) A, the set of terms $A = \bigcup_{b \in B} b.terms$; (2) $T = [\![t_1, t_m]\!]$, the time interval made of m timestamps[2], and (3) $V = \{v_1, ..., v_n\}$, the districts obtained after discretizing the geographical space in n areas.

[1] The posts emitted consecutively in a given period of time.

[2] In our experiments, a timestamp corresponds to an interval of 3 hours.

For example, the geographical area defined by the square $[\min_{b\in B} b.loc.x, \max_{b\in B} b.loc.x] \times [\min_{b\in B} b.loc.y, \max_{b\in B} b.loc.y]$ can be divided along a grid and each square is associated to an element of V. Let area : $\mathbb{R}^2 \to V$ be the function that maps GPS coordinates (x, y) to V. V is hereafter considered as the vertices of a graph $G_V = (V, E)$ whose edges E connect vertices corresponding to adjacent areas.

Our approach is based on the detection of strong variations in term occurrences for a vertex and a timestamp. To this end, we consider the number of occurrences of a term $a \in A$, for a vertex $v \in V$ and a time $t \in T$ as function occ:

$$\text{occ}(a, v, t) = |\{b \in B \mid (a \in b.terms) \text{ and } (\text{area}(b.loc) = v) \text{ and } (b.time = t)\}|.$$

As generally admitted [1, 227], a geolocated event reflects that a large number of messages deal with the same subject during the same time and place. Each topic is associated with the set of its representative terms. For instance, in Fig. 5.13, the first detected event in New York is the *New York Comic Con*, associated with the terms (#nycc, #nycc2016, etc.). These terms burst in a period of three days, and around the location of Jacob K. Javits Convention Center where the convention took place at that time. Thus, a term is likely to correspond to an event in a space-time zone if its number of occurrences is significantly greater than what is observed in the other times for the same space zone. To that end, we compute the mean $\mu(a, v)$ and standard deviation $\sigma(a, v)$ of occ over all the timestamps $[\![t_1, t_m]\!]$. If the difference between $\text{occ}(a, v, t)$ and its average value is greater than θ times the standard deviation, then the number of occurrences of a is said to be significant for v and t. $\theta \geq 0$ is a parameter that controls the sensibility of the method, and whose default value is set to 1. Then, to reduce the impact of very frequent terms, the significance of each term is weighted by the normalized inverse document frequency factor [107]:

$$\text{idf}(a) = 1 - \frac{\log\big(\text{occ}(a, V, T)\big)}{\log\big(\text{occ}(A, V, T)\big)}.$$

Hence, the score function is:

$$\text{score}(a, v, t) = \Big(\text{occ}(a, v, t) - \big(\mu(a, v) + \theta\sigma(a, v)\big)\Big) \times \text{idf}(a),$$

with $\sigma(a, v) = \sqrt{\frac{1}{m-1}\sum_{t\in T}\left(\text{occ}(a, v, t) - \mu(a, v)\right)^2}$ and $\mu(a, v) = \frac{\sum_{t\in T}\text{occ}(a,v,t)}{m}$. A term a is bursting for (v, t) if its score value is positive. The set of bursting terms for a space-time zone (v, t) is thus defined by:

$$\mathcal{A}(v, t) = \{a \in A \mid \text{score}(a, v, t) > 0\}.$$

For the sake of simplicity, given $U \in V$ and $I \sqsubseteq T$, we generalize $\mathcal{A}(U, I) = \bigcap_{v\in U}\bigcap_{t\in I}\mathcal{A}(v, t)$.

We consider a geolocated pattern P as a tuple (U, I, K) where $U \subseteq V$, $I \sqsubseteq T$, and $K \subseteq H$. We aim to identify patterns $P = (U, I, K)$ corresponding to events described by a location U, a time interval I and a set of terms K. The ability for P to represent an event is evaluated by the measure $\mathcal{M}(P)$:

$$\mathcal{M}(P) \quad = \quad \sum_{v\in U}\sum_{t\in I}\sum_{a\in K}\text{score}(a, v, t).$$

The larger $\mathcal{M}(P)$, the higher than expected the frequency of the terms in (U, I). This means that when $\mathcal{M}(P)$ is large, the pattern $P = (U, I, K)$ is likely to be an event. In our problem formulation, we are only interested in patterns $P = (U, I, \mathcal{A}(U, I))$ where $\mathcal{A}(U, I)$ are terms that burst in all the space-time (U, I). This makes it possible to filter out a large part of noisy terms. In fact, it is not likely that a noisy term bursts simultaneously in all the space-times of (U, I). Consequently, in what follows, we are only considering patterns $P = (U, I, \mathcal{A}(U, I))$ and denote them $P = (U, I)$ (since K is uniquely determined by U and I).

In practice, the interest of an event strongly depends on the end-user. Indeed, a user may be much more interested in events related to some subjects (e.g., sport, music) or may prefer events happening in some specific locations (e.g., near her residence). To take user interests into account, we propose to integrate the proper interests of the user through an interactive process. To this end, we define the quality measure $\mathcal{M}_{ud}(P)$ that includes the user's preferences as:

$$\mathcal{M}_{ud}(P) = \sum_{v \in U} \sum_{t \in I} \sum_{a \in \mathcal{A}(U,I)} \text{score}(a, v, t) \times Q_{ud}(a, v),$$

where Q_{ud} increases with the interest of u on a and v. Thus, if P contains some terms a or vertices v that are of interest according to user's feedback, then $\mathcal{M}_{ud}(P)$ increases. When no user feedback is available for a pair (a, v), $Q_{ud}(a, v)$ is equal to 1. We thoroughly discuss in Section 5.4 how the function Q_{ud} is defined.

A user-driven geolocated event must be both spatially compact and have a significant value on its quality measure:

Definition 5.1 (User-driven geolocated event). Given a set of posts B, a set of timestamps T, a graph $G_V = (V, E)$, and a threshold $\delta > 0$, a pattern $P = (U, I)$, with $S \subseteq V$ and $I \sqsubseteq T$ (an interval of T), is a user-driven geolocated event iff (1) $G_V[U]$ is connected and (2) $\mathcal{M}_{ud}(P) \geq \delta$.

If the user does not provide any feedback, the measure $\mathcal{M}_{ud}$ is equal to $\mathcal{M}$. In such particular case, we speak of *data-driven* geolocated events.

Different geolocated events may overlap in terms, geographical area, and timestamps they share, depicting the same real-life event. This has two main disadvantages: (1) the size of the result set may be uselessly very large and redundant, and (2) the method performance may degrade due to the size of the output. Therefore, instead of finding the complete set of events, our goal is to return a concise summary that covers sufficiently all the events. To this end, we use the coverage measure cov. This function, defined for two sets or intervals X and Y, measures how much the set Y covers the set X: $\text{cov}(X, Y) = \frac{|X \cap Y|}{|X|}$. This coverage measure has been used in several problems in order to avoid the redundancy in the results [23, 213]. To measure how much a pattern $P_2 = (U_2, I_2)$ covers a pattern $P_1 = (U_1, I_1)$, we impose that P_2 sufficiently covers P_1 in all the dimensions. The function cover is thus:

$$\begin{aligned} \text{cover}(P_1, P_2) = \; & \min\{\text{cov}(U_1, U_2), \text{cov}(I_1, I_2), \\ & \text{cov}(\mathcal{A}(U_1, I_1), \mathcal{A}(U_2, I_2))\}. \end{aligned}$$

To get a set of patterns that cover all user-driven geolocated events while eliminating some redundancy, we propose to only retrieve a coverage guaranteed event summary.

Definition 5.2 (Coverage guaranteed event summary). Given a threshold $\texttt{minCov} \in [0, 1]$, a coverage guaranteed event summary $\mathcal{R}_1$ of the set of all geolocated events $\mathcal{R}$ fulfills the following property:

$$\forall P \in \mathcal{R},\ \exists P' \in \mathcal{R}_1,\ \text{such that cover}(P, P') \geq \texttt{minCov}.$$

The last drawback that may appear is that two concomitant events in time and space are merged by our approach. To ensure that an event-pattern is semantically coherent – that is its related terms correspond to the same real life event – a post-processing of the event patterns is performed. In Section 5.5.3, we explain how we apply a Louvain clustering [33] on terms to achieve this goal. The similarity between terms is computed based on their co-occurrences, since event-related terms tend to co-occur in the posts.

We propose two approaches to detect geolocated events: SIGLER-Cov that computes a coverage guaranteed event summary $\mathcal{R}_1$, and SIGLER-Samp that samples the pattern space according to the value of $\mathcal{M}_{ud}$. These approaches are formally defined in Section 5.5. The following section details the computation of the user-driver weight used in the quality measure $\mathcal{M}_{ud}(P)$.

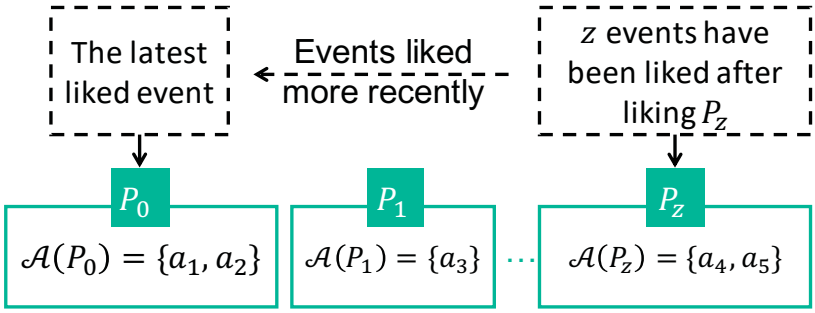
5.4 Integration of user feedback into quality measure

Let us now consider how to incorporate user interest into the geolocated event discovery process. Once a set of events has been identified, a user u appraises the detected events and indicates if she likes it. In doing so, she constructs a partial order on the patterns that is used to favor the discovery of the forthcoming geolocated events toward those of interest for u. User interest is expressed by $Q_{ud}(a, v) = \frac{Q^A_{ud}(G_A,a)+Q^V_{ud}(G_V,v)}{2}$, the average of two interest measures on terms and vertices:

1. $Q^A_{ud} : (G_A, a)$ takes as parameters a weighted graph G_A on terms and a term a. It assigns a value in $[1, \texttt{maxPref}]$, with $\texttt{maxPref} > 1$ a user-defined parameter, based on (1) the neighborhood of a in G_H and (2) a partial order on terms of A derived from the user event ranking. The closer a is to other liked terms in G_A, or the more recently it has been liked, the more $Q^A_{ud}(G_A, a)$ is close to $\texttt{maxPref}$. Otherwise, it tends to 1.

2. $Q^V_{ud} : (G_V, v)$ takes as parameters the graph G_V and a vertex $v \in V$. It takes its value in $[1, \texttt{maxPref}]$ and the closer v is to other liked vertices in G_V, or the more recently it has been liked, the more $Q^V_{ud}(G_V, v)$ is close to $\texttt{maxPref}$.

$\texttt{maxPref} > 1$ represents the maximum value that Q_{ud} can reach. The choice of $\texttt{maxPref}$ depends on how much we want to take into account the user preferences into the process. In the following, we use the value $\texttt{maxPref} = 3$, empirically chosen as the one that maximizes the number of liked events on NYC dataset (see the description of the experiment in section 5.6.3).



a	a_1	a_2	a_3	a_4	a_5
$rank(a)$	0	0	1	z	z

Figure 5.2: Examples of liked events $\{P_1, P_2, ..., P_z\}$, and the values of $rank$ for terms associated to these events.

Both functions $Q^A_{ud}(G_A, h)$ and $Q^V_{ud}(G_V, v)$ are constructed above a weighted graph $G_X = (X, Y, W)$ with X a set of vertices, Y a set of edges and W the function that associates a weight to the edges ($W : Y \to \mathbb{N}$).

- Q^A_{ud} is evaluated on G_A, the term co-occurrence graph, such that $X = A$, $Y = A \times A$ and $\forall a_i, a_j \in A$, $W(a_i, a_j)$ equals the number of posts in which a_i and a_j appear simultaneously:

$$W(a_i, a_j) = |\{b \in B \mid a_i \in b.terms \text{ and } a_j \in b.terms\}|.$$

  Related terms may co-occur in several posts and thus will tend to be connected by a shorter path whose sum of weights is high in G_A.

- Q^V_{ud} is evaluated on the graph G_V, with $X = V$, $Y = E$ and $W : E \to 1$.

In a similar way to PageRank score [166], we valuate the vertices of the graph such that (1) a high score associated to a vertex is transferred to its neighbors, with whom it is connected with a high weight, and (2) this effect decreases all the more as the neighboring vertex is far from the source vertex in the graph. However, unlike PageRank score, which is based on a random walk, we use a concentric model from the vertex to be valued. The vertex score is only influenced by the weights of its neighbors and not by their degrees:
$Q_X(G_X, x) = \alpha \sum_{(x,x') \in Y} \frac{W(x,x')}{\deg(x)} \times Q_X(G_X, x') + (1 - \alpha)\frac{1}{|X|}$, with $\deg(x) = \sum_{(x,x') \in Y} W(x, x')$.
The first term, $\sum_{(x,x') \in Y} \frac{W(x,x')}{\deg(x)} \times Q(G_X, x')$, is simply the weighted average of qualities $Q_X(G_X, x')$ of the neighbor vertices. The second term, $\frac{1}{|X|}$, is a constant corresponding to the probability to directly reach a vertex. $\alpha \in]0, 1[$, whose default value is 0.7, is a balancing parameter between the two terms.

We propose to integrate the user preferences in the second term by replacing the constant value with the function $\mathcal{B}_{ud}(x)$, a weight that amplifies the importance of the vertices involved in recently liked

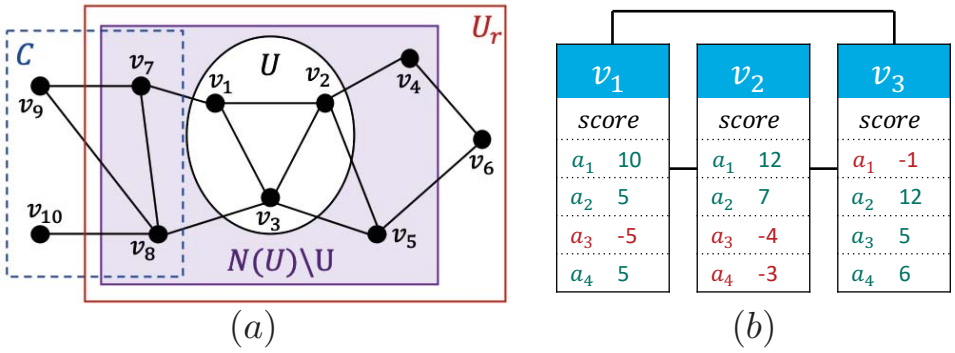


Figure 5.3: A toy example that illustrates (a) the subgraph enumeration, and (b) the scores of terms in some vertices.

events:

$$Q_{X_{ud}}(G_X, x) = \alpha \sum_{(x,x')\in Y} \frac{W(x,x')}{\deg(x)} Q_{X_{ud}}(G_X, x') + (1-\alpha)\mathcal{B}_{ud}(x).$$

$\mathcal{B}_{ud}(x)$ depends on the direct relation between x and the liked events. Let rank(x) be the number of events that have been liked since the last event that (1) contained x and (2) was liked by the user. Fig. 5.2 shows the values of $rank$ for terms that appear in an ordered set of liked events $\{P_0, ..., P_z\}$. P_0 is the latest liked event, thus $rank(h_1) = 0$. The user liked P_z, but she liked z events after that, so $rank(h_5) = z$. Particularly, if a term or a vertex x does not belong to any liked event, then $rank(x) = +\infty$. We define $\mathcal{B}_{ud}(x)$ as $\mathcal{B}_{ud}(x) = 1 + \frac{\texttt{maxPref}-1}{1+log_2(1+\text{rank}(x))}$. We can observe that $\mathcal{B}_{ud}(x) \in [1, \texttt{maxPref}]$, and if $rank(x) = +\infty$, then $\mathcal{B}_{ud}(x) = 1$. It increases if x is related to a recently liked event (if $rank(x) = 0$, $\mathcal{B}_{ud}(x) = \texttt{maxPref}$). The $\log_2$ function is used to smooth the effect of past liked events.

$Q_{X_{ud}}(G_X, x)$ can be rewritten as the matrix equation $A.Q_{X_{ud}} = B$ with (1) $a_{ij} = 1$ if $i = j$, and $a_{ij} = -\alpha\frac{W(x_i,x_j)}{\deg(x_i)}$ otherwise; (2) $b_i = (1-\alpha)\mathcal{B}_{ud}(x_i)$. This equation can be solved thanks to the Jacobi method, as the convergence condition below is satisfied.

Proposition 5.1. The matrix A is strictly diagonally dominant.

Proof. As $\sum_{i\neq j}|a_{ij}| = \alpha\sum_{i\neq j}\frac{W(x_i,x_j)}{\deg(x_i)} = \alpha < 1$ and $|a_{ii}| = 1$, we have $|a_{ii}| > \sum_{i\neq j}|a_{ij}|$. □

5.5 Computing geolocated events

We first present SIGLER-Cov, an algorithm that computes a set of patterns $\mathcal{R}_1$ whose quality $\mathcal{M}_{ud}$ exceeds the threshold δ. In addition, $\mathcal{R}_1$ is a coverage guaranteed event summary (see Definitions 5.1 and 5.2). Then, we present SIGLER-Samp, a sampling event detection approach. Finally, we explain the post-processing step as introduced before.

5.5.1 Event Detection with Coverage Guarantee

We propose to extract user-driven geolocated events using a generate and test approach. It first enumerates a time interval I, and then explores the connected subgraphs S corresponding to areas where posts were sent during I. The quality measure $\mathcal{M}_{ud}(U, I)$ evaluates whether the terms of the posts sent from $G_V[U]$ during I are characteristic of this time frame. As the number of such generated events is very large, especially the number of possible subgraphs, we propose several pruning techniques that makes the extraction feasible on real-life generated sets of posts. SIGLER-Cov (Algorithm 6) enumerates all the intervals $[t_i, t_j]$ included in $T = [\![t_1, t_m]\!]$ thanks to the two loops in lines 2 and 4. For each interval I, it explores the connected subgraphs of $G_V[C]$ – the graph induced by the vertices for which there exists at least a term that bursts during the interval I – calling the function SGEnumerate presented in Algorithm 7. This backtracking algorithm uses two sets of vertices: $U \subseteq V$, the current enumerated subgraph induced by U, and $C \subseteq V \setminus U$ the vertices that are still to be explored. Initially, $U = \emptyset$ and C contains all the vertices for which there exists a bursting term in I. Algorithm 7 enumerates vertices from $C \cap N(U)$, with $N(U)$ the neighbors of vertices of U: $N(U) = \{v \in V \mid \exists u \in U \ : \ (u, v) \in E\}$. Once the candidate set C is empty, P is tested to only be retained if it satisfies definitions 5.1 and 5.2 (lines 11 to 15). Fig. 5.3 (a) shows an example of sets U, $N(U)$ and C. In the next step of this example, SGEnumerate will choose v_7 or v_8 (in the intersection of C and $N(U)$) and add it to U. We can notice that some vertices do not belong to C (e.g., v_4) because they have already been enumerated and removed from C.

Algorithm 6: SIGLER-Cov(δ, `minCov`, $\mathcal{R}_1$)

Input: δ the quality threshold, `minCov` the coverage threshold
Output: $\mathcal{R}_1$ a set of coverage guaranteed geolocated events

1 $\mathcal{R}_1 \leftarrow \emptyset$
2 **for** $i \leftarrow 1$ *to* m **do**
3 $C \leftarrow V$
4 **for** $j \leftarrow i$ *to* m **do**
5 $I \leftarrow [t_i, t_j]$
6 $C \leftarrow \{v \in C \mid \mathcal{A}(\{v\}, I) \neq \emptyset\}$
7 SGEnumerate(I, $\emptyset$, C,$\mathcal{R}_1$, δ, `minCov`)

We use three pruning mechanisms without which the algorithm does not scale. The first one – the anti-monotonicity of $\mathcal{A}(P)$ – is used line 6 of Algorithm 6 to prune vertices given a time interval. This property is defined up to the intuitive partial order $\subseteq$: $P_1 \subseteq P_2$ if and only if $U_1 \subseteq U_2$ and $I_1 \subseteq I_2$.

Proposition 5.2 (anti-monotony of $\mathcal{H}(P)$). Considering two patterns $P_1 = (U_1, I_1)$ and $P_2 = (U_2, I_2)$, if $P_1 \subseteq P_2$ then $\mathcal{A}(P_2) \subseteq \mathcal{A}(P_1)$.

Proof. If $a \in \mathcal{A}(P_2)$, then $\forall v \in U_2, \forall t \in I_2$, score$(a, v, t) > 0$. As $U_1 \subseteq U_2$ and $I_1 \subseteq I_2$, thus, $\forall v \in U_1, \forall t \in I_1$, score$(a, v, t) > 0$, and thus $a \in \mathcal{A}(P_1)$. □

In Fig. 5.3 (b), the bursting terms of $U' = \{v_1\}$ are: $\mathcal{A}(U', t) = \{a_1, a_2, a_4\}$. If U' is expanded with a vertex x, the new set $\mathcal{A}(U' \cup \{x\}, t)$ is necessarily included in the previous one. For example,

Algorithm 7: SGEnumerate(I, U, C, $\mathcal{R}_1$, δ,minCov)

Input: I a time interval, U the current explored subgraph, C the candidate sets, δ the quality threshold, minCov the coverage threshold

Output: $\mathcal{R}_1$ the set of events

```
1  P ← (U, I)
2  if C ∩ N(U) ≠ ∅ then
3      if UB(S ∪ C, I, A(P)) ≥ δ then
4          for P_r ∈ R_1 do
5              if LB(P, C, P_r) ≥ minCov then
6                  return
7          Choose a vertex v ∈ C ∩ N(U)
8          SGEnumerate(I, U ∪ {v}, C \ {v}, δ, minCov)
9          SGEnumerate(I, U, C \ {v}, δ, minCov)
10 else
11     if M_ud(P) ≥ δ then
12         for P_r ∈ R_1 do
13             if cover(P, P_r) ≥ minCov then
14                 return
15         R_1 ← R_1 ∪ {P}
```

$\mathcal{A}(\{v_1, v_2\}, t) = \{a_1, a_2\}$.

The second pruning mechanism is based on the computation of an upper-bound of $\mathcal{M}_{ud}(P)$ (used line 3 in Algorithm 7).

Definition 5.3 (Upper-bound on $\mathcal{M}_{ud}$). Let $U \subseteq V$, $I \sqsubseteq T$, and $J \subseteq A$. UB is defined as:

$$UB(U, I, J) = \sum_{v \in U} \sum_{t \in I} \sum_{a \in J} \max\big(\text{score}(a, v, t) \times Q_{ud}(a, v), 0\big).$$

We denote by $\Gamma(I, U, C)$ the set of all patterns that can be reached when expanding U by adding vertices from C: $\Gamma(I, U, C) = \{(U', I) \mid U' \subseteq U \cup C \text{ and } U \subseteq U'\}$, that is to say, the patterns that are generated from SGEnumerate(I, U, C, $\mathcal{R}_1$, δ, minCov). The following property states that $UB(U \cup C, I, \mathcal{A}(U, I))$ upper bounds $\mathcal{M}_{ud}$ for all subsequent patterns:

Proposition 5.3. Let $U \subseteq V$, $I \sqsubseteq T$, and $C \subseteq V \setminus S$. For all patterns $P' = (S', I) \in \Gamma(I, U, C)$, $UB(U \cup C, I, \mathcal{A}(U, I)) \geq \mathcal{M}_{ud}(U', I)$.

Proof. Since $U' \subseteq U \cup C$ and $\mathcal{A}(P') \subseteq \mathcal{A}(P)$, we have

$$\begin{aligned}
&UB(U \cup C, I, \mathcal{A}(P)) = \\
&\sum_{v \in U'} \sum_{t \in I} \sum_{a \in \mathcal{A}(P')} \max\big(\text{score}(a, v, t) \times Q_{ud}(a, v), 0\big) \\
&+ \sum_{v \in U \cup C} \sum_{t \in I} \sum_{a \in \mathcal{A}(P) \setminus \mathcal{A}(P')} \max\big(\text{score}(a, v, t) \times Q_{ud}(a, v), 0\big) \\
&+ \sum_{v \in U \cup C \setminus S'} \sum_{t \in I} \sum_{a \in \mathcal{A}(P')} \max\big(\text{score}(a, v, t) \times Q_{ud}(a, v), 0\big),
\end{aligned}$$

$$\geq \sum_{v \in U'} \sum_{t \in I} \sum_{a \in \mathcal{A}(P')} \max\big(\text{score}(a,v,t) \times Q_{ud}(a,v), 0\big),$$
$$\geq \sum_{v \in U'} \sum_{t \in I} \sum_{a \in \mathcal{A}(P')} \text{score}(a,v,t) \times Q_{ud}(a,v) = \mathcal{M}_{ud}(P').$$

The last pruning technique is built on the coverage measure. As stated in Definition 5.2, there may exist several coverage guarantee summaries of geolocated events. Whereas it might be interesting to have a summary of smallest cardinality, the problem of finding the set of minimal size is NP hard. A practical approach consists in constructing the summary during the enumeration. We also use the coverage measure to prune large parts of the subgraph search space thanks to a lower bound LB (lines 4 to 6 in Algorithm 7) that will be defined next. The intuition behind is to prune a search space $\Gamma(I,U,C)$ if it is covered by an already found pattern P_r.

Proposition 5.4. Given a geolocated event pattern $P_r = (U_r, I_r)$ and a pattern $P' = (U', I)$ in $\Gamma(I,U,C)$, we have $\text{cov}(U', U_r) \geq \text{cov}(U \cup (C \setminus U_r), U_r)$.

Proof. As $U' = U \cup C'$ s.t $C' \subseteq C$, we have:

$$cov(U', U_r) = \frac{|U' \cap U_r|}{|U'|} = \frac{|U \cap U_r| + |C' \cap U_r|}{|U| + |C' \setminus U_r| + |C' \cap U_r|} \geq \frac{|U \cap U_r| + |C' \cap U_r|}{|U| + |C \setminus U_r| + |C' \cap U_r|},$$

We define $g(x) = \frac{|U \cap U_r| + x}{|U| + |C \setminus U_r| + x}$, with $x = |C' \cap U_r|$. Knowing that the derivative $g'(x) = \frac{|U| + |C \setminus U_r| - |U \cap U_r|}{(|U| + |C \setminus U_r| + x)^2} \geq 0$, $g(x)$ takes its lower value when x is minimal, that is when $x = 0$. Thus:

$$g(x) \geq \frac{|U \cap U_r|}{|U| + |C \setminus U_r|} = cov(U \cup (C \setminus U_r), U_r),$$

We conclude that $cov(U', U_r) \geq cov(U \cup (C \setminus U_r), U_r)$.

For instance, in Fig. 5.3, U_r covers U and a part of C. $\text{cov}(U \cup \{v_9, v_{10}\}, S_r) = \frac{3}{5}$ is a lower bound of $\text{cov}(U', U_r)$ for all $(U', I) \in \Gamma(I,U,C)$. In fact, $U \cup \{v_9, v_{10}\}$ contains only the vertices of C that are not in U_r, which minimizes the coverage of U_r.

Definition 5.4. Let $\mathcal{A}(P) \cap \mathcal{A}(Pr) = \{a_1, \ldots, a_q\}$ – the set of terms of P covered by those of P_r – be ordered by $a_i \leq_{UB} a_j$ iff $UB(U \cup C, I, \{a_i\}) \geq UB(U \cup C, I, \{a_j\})$. We consider the minimal set $\{a_1, \ldots, a_{q^\star}\}$ of terms that can be added to $\mathcal{A}(P) \setminus \mathcal{A}(P_r)$ while still satisfying the upper-bound:

$$q^\star = \underset{r \in \{1 \ldots q\}}{\text{argmin}}\ UB(U \cup C, I, \mathcal{A}(P) \setminus \mathcal{A}(P_r) \cup \{a_1, \cdots, a_r\}) \geq \delta$$

and define $\mathcal{A}^\star$ as $\mathcal{A}(P) \setminus \mathcal{A}(P_r) \cup \{a_1, \cdots a_{q^\star}\}$. In other words, $\mathcal{A}^\star \subseteq \mathcal{A}(P)$ is the set of terms that overlaps the least with $\mathcal{A}(P_r)$ while verifying the condition $UB(U \cup C, I, \mathcal{A}^\star) \geq \delta$

Proposition 5.5. For each pattern $P' = (U', I) \in \Gamma(I,U,C)$ such that $\mathcal{M}_{ud}(P') \geq \delta$, we have $\text{cov}(\mathcal{A}(P'), \mathcal{A}(P_r)) \geq \text{cov}(\mathcal{A}^\star, \mathcal{A}(P_r))$.

Proof. We know that $|\mathcal{A}(P') \cap \mathcal{A}(P_r)| \geq q^\star$, otherwise $UB(U', I, \mathcal{A}(P')) < \delta$ and $\mathcal{M}_{ud}(P') < \delta$. We can rewrite $cov(\mathcal{A}(P'), \mathcal{A}(P_r)) = \frac{|\mathcal{A}(P') \cap \mathcal{A}(P_r)|}{|\mathcal{A}(P') \setminus \mathcal{A}(P_r)| + |\mathcal{A}(P') \cap \mathcal{A}(P_r)|} = \frac{q^\star + x}{|\mathcal{A}(P') \setminus \mathcal{A}(P_r)| + q^\star + x}$ with $x \geq 0$. Let's denote $g(x) = \frac{q^\star + x}{|\mathcal{A}(P) \setminus \mathcal{A}(P_r)| + q^\star + x}$. We have $cov(\mathcal{A}(P'), \mathcal{A}(P_r)) \geq g(x)$ as $\mathcal{A}(P') \subseteq \mathcal{A}(P)$. Knowing that the derivative $g'(x) = \frac{|\mathcal{A}(P) \setminus \mathcal{A}(P_r)|}{(|\mathcal{A}(P) \setminus \mathcal{A}(P_r)| + q^\star + x)^2} \geq 0$, then $g(x)$ takes its lower value when $x = 0$. Thus, $g(x) \geq \frac{q^\star}{|\mathcal{A}(P) \setminus \mathcal{A}(P_r)| + q^\star} = cov(\mathcal{A}^\star, \mathcal{A}(P_r))$ and we conclude the proof. □

Definition 5.5. We define the function LB as

$$\begin{aligned} LB(P, C, P_r) \quad = \quad & \min\{\text{cov}(U \cup (C \setminus U_r), U_r), \text{cov}(I, I_r), \\ & \text{cov}(\mathcal{A}^\star, \mathcal{A}(P_r))\} \end{aligned}$$

Proposition 5.6. For each pattern $P' = (U', I) \in \Gamma(I, U, C)$ such that $\mathcal{M}_{ud}(P') \geq \delta$, we have $\text{cover}(P', P_r) \geq LB(P, C, P_r)$.

Proof. It is sufficient to prove that : $cov(U', U_r) \geq cov(U \cup (C \setminus U_r), U_r)$ and $cov(\mathcal{A}(P'), \mathcal{A}(P_r)) \geq cov(\mathcal{A}^\star, \mathcal{A}(P_r))$ and $cov(I, I_r) \geq cov(I, I_r)$ The first two equations are respectively guaranteed with Propositions 5.4 and 5.5, and the third one is trivial. □

The exploration of the search space $\Gamma(I, U, C)$ is stopped if there exists $P_r \in \mathcal{R}_1$ such that $LB(P, C, P_r) \geq minCov$, because all the subsequent patterns are covered by it.

5.5.2 Pattern Sampling Based Event Detection

In this section, we explore another approach to discover geolocated events: Pattern sampling [30]. Given a time budget, the proposed algorithm SIGLER-Samp mines events using a random exploration of the search space that favors events with high $\mathcal{M}_{ud}$ values. Such an approach enables instant mining which is required in interactive pattern mining.

The pattern sampling process we consider is based on a random walk on a graph whose vertices are patterns $P = (U, I)$ and edges (transitions) are chosen following a probability measure that overweights high quality patterns. The random walk starts from a singleton pattern $P = (U, I)$ where $S = \{v\}$ and $I = [t, t]$. Next, P is expanded by adding randomly drawn vertices from $N(U) \setminus U$, the neighborhood of U, or by adding a timestamp to I. The random walk is basically composed of two main steps described in Algorithm 8:

1. $\mathcal{M}_{ud}(P)$ is computed for each pattern $P = (U, I)$, where $U = \{v\},\ v \in V$ and $I = [t, t],\ t \in T$. The probability of drawing a singleton pattern P is defined as $\mathcal{P}(P) = \frac{\mathcal{M}_{ud}(P)}{\sum_{v' \in V, t' \in T} \mathcal{M}_{ud}(\{v'\}, [t', t'])}$
2. From a current pattern $P = (U, I)$ with $I = [t_i, t_j]$, the next step consists to draw a pattern P' from the set:

$$\begin{aligned} \text{Next}(P) \quad = \quad & \{(U, I)\} \cup \{(U, [t_{i-1}, t_j])\} \cup \{(U, [t_i, t_{j+1}])\} \\ \cup \quad & \{(U', I) \mid U' = U \cup \{v\},\ v \in N(U) \setminus U\} \end{aligned}$$

This is done based on $\mathcal{P}(P'|P)$, the probability to reach the pattern $P' \in \text{Next}(P)$ from P: $\mathcal{P}(P'|P) = \frac{\mathcal{M}_{ud}(P')}{\sum_{P_2 \in \text{Next}(P)} \mathcal{M}_{ud}(P_2)}$.

This distribution of probabilities rewards transitions toward patterns P' with large $\mathcal{M}_{ud}(P')$ values. After drawing P', if $P' \neq P$, we continue the expansion by repeating Step 2 using the new pattern P'. If $P' = P$, the pattern P is returned to the user and the sampling is repeated from Step 1 until the whole consumption of the time budget.

The two main steps are repeated (lines 8 to 21) until $P' = P$. At each iteration, the pattern $P = (U, I)$ is extended by adding a vertex to U or a timestamp to I. Since U and I are respectively bounded by V and T, this loop necessarily stops after at most $|V| + |I|$ iterations. All patterns with nonzero $\mathcal{M}_{ud}$ value have a non zero probability to be generated.

Proposition 5.7. For each pattern $P = (U, I)$, if $\mathcal{M}_{ud}(P) > 0$ then $\mathcal{P}(P \in \mathcal{R}_2) > 0$

Proof. Let us prove it by induction on $n = |U| + |I|$.

- For $n = 2$, P is such that $|U| = 1$ and $|I| = 1$ (in other cases, $\mathcal{M}_{ud}(P) = 0$). P can be drawn in the first step, and if it is chosen from Next(P) in step 2, it is added to $\mathcal{R}_2$. Thus, $\mathcal{P}(P \in \mathcal{R}_2) \geq \frac{\mathcal{M}_{ud}(P)}{\sum_{v' \in V, t' \in T} \mathcal{M}_{ud}(\{v'\},[t',t'])} \times \frac{\mathcal{M}_{ud}(P)}{\sum_{P_2 \in \text{Next}(P)} \mathcal{M}_{ud}(P_2)} > 0$.
- Let us suppose that the proposition is true for n. Let $P = (U, I)$ be a pattern such that $|U| + |I| = n + 1$ and $\mathcal{M}_{ud}(P) > 0$. Let $P' = (U', I') \subseteq P$ be a pattern containing one less vertex or one less timestamp than P. This means that $|U'| + |I'| = n$, by the recursion hypothesis we have $\mathcal{P}(P' \in \mathcal{R}_2) > 0$. Thus, the probability $\mathcal{P}(P')$ to reach P' is not null. Since P can be reached from P' during the random walk, then: $\mathcal{P}(P \in \mathcal{R}_2) \geq \mathcal{P}(P') \times \frac{\mathcal{M}_{ud}(P)}{\sum_{P_2 \in \text{Next}(P')} \mathcal{M}_{ud}(P_2)} \times \frac{\mathcal{M}_{ud}(P)}{\sum_{P_2 \in \text{Next}(P)} \mathcal{M}_{ud}(P_2)}$
  $\mathcal{P}(P \in \mathcal{R}_2) > 0$

□

Fig. 5.4 gives an example of an iteration in SIGLER-Samp. The current generated pattern is $P = (\{v_4\}, [t_3, t_4])$. In the next iteration, one of the neighbors of P, or P, is randomly chosen. The neighbors are generated by either adding $\{v_2\}$ or $\{v_6\}$, vertices connected to v_4 in the graph of Fig. 5.3, or increasing the time interval.

5.5.3 Discussion

We discuss here some potential issues that may appear, and the related post-processing to fix them. By applying one of the aforementioned algorithms, we find the set of patterns $\mathcal{R}_1$ or $\mathcal{R}_2$, let's denote it $\mathcal{R}_*$. In some particular cases, two real life events happen at the same time and the same location. However, these two events will be merged in the same pattern $P \in \mathcal{R}_*$. It is important to separate them before displaying the result to the end user. Therefore, for each pattern $P = (U, I, \mathcal{A}(U, I)) \in \mathcal{R}_*$, we apply a

Algorithm 8: SIGLER-Samp

Input: `time_budget`
Output: $\mathcal{R}_2$ a set of sampled patterns

for $v \in V, t \in T$ **do**
  compute $\mathcal{M}_{ud}(\{v\}, [t,t])$
while *current_time* $<$ `time_budget` **do**
  // Step 1: draw a singleton pattern P
  draw $P = (\{v\}, [t,t]) \sim \frac{\mathcal{M}_{ud}(P)}{\sum_{v' \in V, t' \in T} \mathcal{M}_{ud}(\{v'\},[t',t'])}$
  // Step 2: expansion of P
  $P' \leftarrow P$
  repeat
    $P \leftarrow P'$
    // Compute the set Next(P) for $P = (U, [t_i, t_j])$
    Next$(P) \leftarrow \{P\}$
    for $v \in N(U) \setminus U$ **do**
      Next$(P) \leftarrow$ Next$(P) \cup \{(U \cup \{v\}, [t_i, t_j])\}$
    if $i > 1$ **then**
      Next$(P) \leftarrow$ Next$(P) \cup \{(U, [t_{i-1}, t_j])\}$
    if $j < m$ **then**
      Next$(P) \leftarrow$ Next$(P) \cup \{(U, [t_i, t_{j+1}])\}$
    for $P' \in$ *Next*(P) **do**
      compute $\mathcal{M}_{ud}(P')$
    draw $P' \sim \frac{\mathcal{M}_{ud}(P')}{\sum_{P_2 \in \text{Next}(P)} \mathcal{M}_{ud}(P_2)}$
  until $P' = P$;
  $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \cup P$

community detection algorithm on the terms $\mathcal{A}(U, I)$ in order to partition them into groups $K_1, ..., K_d$ where (1) $\forall i \in [\![1, d]\!] : K_i \subseteq \mathcal{A}(U, I)$ (2) $\cup_i K_i = \mathcal{A}(U, I)$ (3) each K_i corresponds to a single real life event. We use Louvain community detection algorithm [33], and we express its result by the function $Louvain_1(P) = \{K_1, ..., K_d\}$. The similarity measure sim_1 used in this clustering is defined for two terms $a_1, a_2 \in \mathcal{A}(U, I)$ w.r.t the pattern P:

$$sim_1(a_1, a_2, P) = \begin{cases} 1, \text{if } |\{b \in B \mid (\{a_1, a_2\} \subseteq b.terms) \text{ and} \\ \quad (\text{area}(b.loc) \in U) \text{ and } (b.time \in I)\}| \geq 1 \\ 0, \text{otherwise} \end{cases}$$

In other words, $sim_1(a_1, a_2, P) = 1$ if a_1 and a_2 co-occur at least once in the space S and the time interval I. Thus, each cluster $K_i \in Louvain_1(P)$ would be a set of terms that co-occur in this space-time. Each cluster K_i gives a pattern (U, I, K_i) with the same space-time than the current P. After applying this clustering to each $P \in \mathcal{R}_*$, we have the post-processed result $R' = \cup_{(U,I,\mathcal{A}(U,I)) \in \mathcal{R}_*} \{(U, I, K) \mid K \in Louvain_1(U, I, \mathcal{A}(U, I))\}$

In order to deal with the redundancy issue, we have defined SIGLER-Cov that computes a summary $\mathcal{R}_1$. However, it is not necessarily the optimal summary, that is to say the summary of minimal size whose events partially cover each geolocated events that does not belong to the summary. Indeed, the

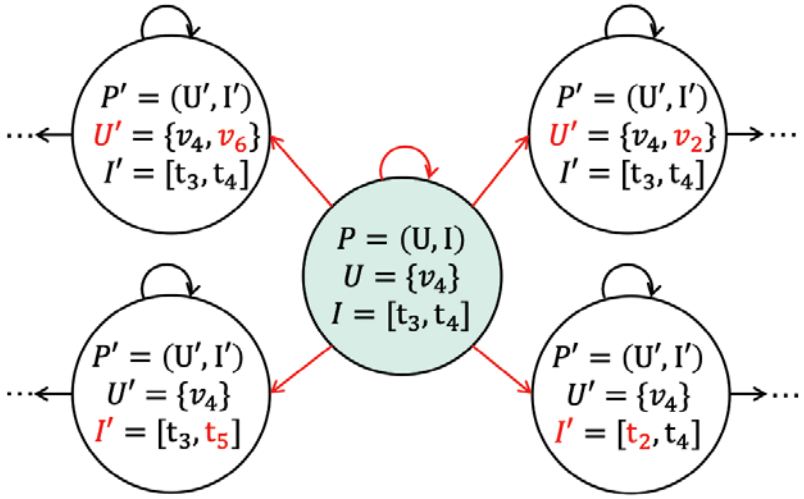


Figure 5.4: Examples of the pattern sampling approach: P' are all the patterns that can be generated from the current pattern P when considering the graph of Fig. 5.3.

computation of an optimal summary is NP hard. Thus, some redundancy can still remain in the result, we post-process the set R' to fix this issue. We apply Louvain algorithm on patterns $P \in R'$ to merge the ones with similar location, time interval, and terms. We define a similarity measure sim_2 for two patterns $P = (U, I, K)$ and $P' = (U', I', K')$: $sim_2(P, P') = \frac{|U \cap U'|}{|U \cup U'|} \times \frac{|I \cap I'|}{|I \cup I'|} \times \frac{|K \cap K'|}{|K \cup K'|}$. The result will be the communities: $\mathcal{C}_1, ..., \mathcal{C}_l$ where each community $\mathcal{C}_i \subseteq R'$ is a set of similar patterns. From each community $\mathcal{C}_i$ we reconstitute a pattern $P_{\mathcal{C}_i} = \cup_{(U,I,K) \in \mathcal{C}_i}(U, I, K)$. The final result set of pattern is: $R'' = \{P_{\mathcal{C}_1}, ..., P_{\mathcal{C}_l}\}$.

5.6 Experiments

In this section, we report our experimental results. We start by describing the real-world datasets we use, as well as the questions we aim to answer. Then, we provide a thorough comparison with the state-of-the-art algorithms and we report a performance study. Eventually, we evaluate the ability of our approach to take user interests into account through different testbeds[3].

SIGLER-Cov is implemented in C++ and the experiments were executed on a machine equipped with i7 CPU @ 2.5GHz, and 16GB main memory, running macOS Sierra version 10.12.2. For reproducibility purposes, the source code and the data are available[4].

[3] We report experiments performed on a crowd-sourcing platform with real-users in this chapter. Additional experiments with virtual users are reported in supplementary material.

[4] `https://goo.gl/1Ptw9J`

5.6.1 Experimental Setting

Experiments are performed on 3 real-world datasets that contain the tweets obtained by querying three different cities on Twitter: New York (NYC), Los Angeles (LA) and London. For each city, we collected geolocated public tweets and removed those produced by bots (i.e, accounts that produce more than 100 tweets in a period of 10 days). We only retained tweets containing hashtags or user mentions. The main characteristics of these datasets are given in Table 5.1.

dataset	starting date	ending date	# tweets	# distinct terms
New York	2016/10/08	2017/01/07	652,244	332,618
Los Angeles	2017/05/17	2017/07/27	353,541	224,769
London	2017/05/17	J 2017/07/27	270,648	177,166

Table 5.1: Description of the real-world datasets.

This empirical study aims to answer the following questions: Are SIGLER-Cov and SIGLER-Samp more effective and efficient than their competitors? Do they scale well according to the size of the dataset and the different parameters? Does SIGLER-Samp capture all the events? Is the approach able to make use of user feedback to discover user-relevant events?

In the first bunch of experiments, we compare our approach with two local event detection approaches: (1) Multiscale Event Detection algorithm (MED) [67], a state-of-the-art algorithm which aims to identify geolocated events based on a wavelet analysis of time series of terms, (2) GeoBurst [227], an online local event detection method, that first detects geo-topic clusters using a random walk on a keyword co-occurrence graph, and then ranks all the clusters with a weighted combination of spatial and temporal burstiness. We also considered Insight [115] and EDCoW [218], two non location-aware event detection methods. Insight is one of the best methods to detect events in tweets, as it won a recent challenge [167].

These experiments show that (1) non location-aware approaches are not appropriate to detect geolocated events, and (2) MED and GeoBurst algorithms are less robust to noise than our approaches, and encounter scalability issue. Finally, we demonstrate the ability of our methods to extract interesting user-driven events via experiments performed on a crowdsourcing platform.

5.6.2 Effectiveness

Evaluation on synthetic data. We first study the ability of our approach to detect user-driven geolocated events. Using the method described in [67], we generate artificial datasets for which the ground-truth geolocated events are known. Twenty events, denoted hereafter R_0, are artificially created. Each of them lasts between 2 and 8 timestamps, involves from 2 to 16 vertices and is defined by 10 unique terms. The datasets contain 1024 vertices, 32 timestamps and 1200 unique terms. Posts are artificially produced and sent at different timestamps and spatial locations. Each post contains between 9 to 13 terms. These posts are either related to embedded events, or are randomly drawn: (1) Event-related posts are uniformly distributed over the vertices and times stamps of its associated event and contain 5 of the 10 event-related

terms as well as between 4 to 8 other terms; (2) Non-event related posts are uniformly distributed over the other timestamps and vertices. The terms they contain are drawn using a Zipf law probability distribution [172] among the 1000 non-event related terms. 10 to 50 event-related posts are randomly drawn and the number of non-event related posts is controlled by the $noise\_rate$ parameter.

Let $R = \{e_1, \ldots, e_k\}$ be the set of discovered events. The quality of R is assessed based on the following adapted $Fscore$ measure:

$$Fscore(R, R_0) \quad = \quad 2 \times \frac{Precision(R, R_0) \times Recall(R, R_0)}{Precision(R, R_0) + Recall(R, R_0)},$$

with $Precision(R, R_0) = \frac{\sum_{e \in R} max_{e' \in R_0} \text{cov}(e,e')}{|R|}$ and $Recall(R, R_0) = \frac{\sum_{e \in R_0} max_{e' \in R} \text{cov}(e,e')}{|R_0|}$.

Fig. 5.5 presents the $Fscore$ values achieved by the different approaches when noise rate is varying. From this figure we can draw the following conclusions:

1. *Non location-aware event detection approaches cannot detect geolocated events*, as INSIGHT and EDCoW $Fscore$ is always lower than 0.2.

2. SIGLER-Cov, MED and GeoBurst perform well with low noise rate. However, *both SIGLER-Cov and SIGLER-Samp are more robust to noise than MED and GeoBurst* . The $Fscore$ of our approaches remains high even when the noise rate increases, whereas MED and GeoBurst $Fscore$ decreases almost linearly. More precisely, MED and GeoBurst keep a good recall but the precision decreases when the noise rate increases.

3. Furthermore, the $Fscore$ of SIGLER-Samp is obviously bounded by the $Fscore$ obtained by SIGLER-Cov which adopts a more exhaustive exploration of the search space. Nevertheless, the bigger the time budget, the better the $Fscore$.

We also study the impact of parameters of our approach on the value of $Fscore$. To this aim, we variate `minCov` and δ and we show the value of $Fscore$ in Fig 5.6. The highest $FScore$ is achieved with values of δ between 1 and 2, this leads to the best trade-off between precision and recall. Increasing δ allows to increase the precision, but decreases the recall. Although the value of $Fscore$ slightly increases when `minCov` is higher, this parameter does not significantly impact the quality of the result.

Evaluation on real data. Based on synthetic data, we have previously studied the ability of our approach to detect local events, and we compared it with other state-of-the-art methods. Ideally, one would prefer to use real world datasets to perform such evaluation. However, we do not have the ground truth of the studied real world data. This makes it very hard to achieve an objective comparison on them. Nevertheless, we show in Table 5.2 the top 10 events returned by SIGLER-Cov, MED, and GeoBurst, on the first 10k tweets of NYC dataset, and we make a discussion about them. The number of tweets is limited to only 10k, in order to be able to compare with MED which has scalability issues.

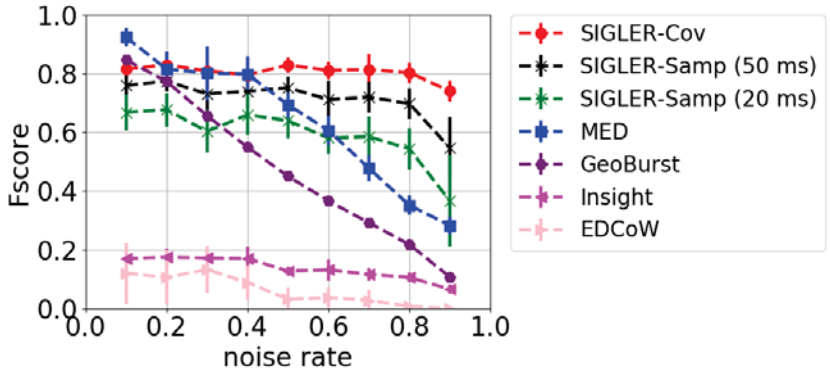


Figure 5.5: Average and error bars of $Fscore$ values obtained over 10 generated datasets for a given noise rate value ($\delta = 5$ and $\texttt{minCov} = 0.8$). Non location-aware event detection approaches are not able to detect geolocated events. MED fails in presence of noise. Runtime of SIGLER-Cov increases from 10ms to 2.5s when the noise rate increases.

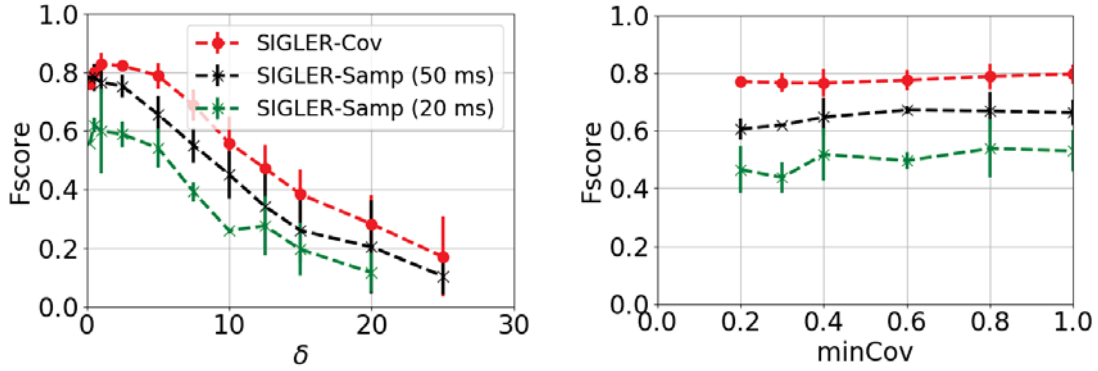


Figure 5.6: Average and error bars of $Fscore$ obtained according to δ and $\texttt{minCov}$. Default values are: $noiseRate = 0.8$, $\texttt{minCov} = 0.8$, $\delta = 5$.

We can notice that there are some similar results of SIGLER-Cov with those of MED and GeoBurst. In fact, SIGLER-Cov and MED have both returned the New York Comic Con[5] (1, 4 and 8 in SIGLER-Cov, 3 and 4 in MED), Beyoncé Concert[6] (2 in SIGLER-Cov, 5 and 8 in MED), and Taylor Mac concert[7] (5 in SIGLER-Cov and 10 in MED) . Both SIGLER-Cov and GeoBurst identified the FOLD Festival of Nile Rodgers[8] (9 in SIGLER-Cov and 1 in GeoBurst). However, the rest of top results of GeoBurst are different from the ones of other approaches. GeoBurst seems to give more importance to small events. In fact, each of the top 10 events of GeoBurst contain at most 10 tweets, while the number of tweets in top results of SIGLER-Cov (resp. MED) varies between 43 and 3k (resp. between 42 and 1280).

[5] https://goo.gl/BR7kgp
[6] https://goo.gl/FrZEBu
[7] https://goo.gl/9pM6z3
[8] https://goo.gl/1htnhk

#	SiglerCov		MED		GeoBurst	
	time	top 5 terms	time	top 5 terms	time	top 5 terms
1	0h to 23h	**nycc, nycc2016, cosplay, ny_comic_con, newyorkcomiccon.**	0h to 23h	nyc, newyork, love, manhattan, saturday.	15h to 23h	**nilerodgers, foldfest, bettemidler, foresthillsstadium, walkeratconcert.**
2	0h to 21h	**formationworldtour, beyonce, formationtour, beyhive, theformationworldtour.**	0h to 23h	newyork, job, hiring, newyorkcity, photo.	15h to 21h	**raniahatoum, thelondonnyc, tripleb, blackbridalbliss, bridalgown.**
3	6h to 18h	**rnrbrooklyn, runrocknroll, halfmarathon, brooklynwerunhard.**	0h to 23h	**nycc2016, cosplay, nycc, newyorkcomiccon, wonderwoman.**	18h to 23h	**intercoiffure, wella, icamoments, wellapro, nerolisalonspa.**
4	12h to 21h	**smashingnycc, nigelthornberry, nigel, smashing, wildthornberries.**	0h to 23h	**nycc, ny_comic_con, cosplay, comiccon, marvel.**	9h to 15h	ridetheferry.
5	12h to 21h	**24decadehistoryofpopularmusic, sawtaylormac, 24decades, marskado, afraidoffun.**	0h to 23h	**formationworldtour, beyonce, beyhive, metlifestadium, beyonce.**	18h to 23h	**sturgillsimpson, kingsbklyn, asailorsguidetoearth.**
6	15h to 23h	**greenday, websterhall, revrad, saturdaynight, 90s.**	0h to 23h	**brooklyn, bushwick, williamsburg,sigurros, music.**	0h to 12h	**elitefridays, cityscapesny, imsobx, reposting, cityscapesnyc.**
7	18h to 21h	**dosgualas, livvinyl, monies, freeze, megaman.**	0h to 23h	repost, montanoy27, regram, alofokemusicnet, parkslopemoms_.	18h to 23h	**descendents.**
8	12h to 18h	**ronswwadventure, 75thanniversary.**	0h to 23h	**formationtour, beyonce, theformationworldtour, nj, kendricklamar.**	0h to 15h	**50cent, dozadrumdealer, mynameisjuan, industrykilla, narcotechs.**
9	15h to 21h	**foresthillsstadium, nilerodgers, fold, bettemidler, foresthills.**	3h to 23h	foodporn, food, foodie, yummy, eeeeeats.	0h to 12h	**deadrabbitnyc.**
10	18h to 21h	**knicks, nets, nyknicks, brooklynnets, preseason.**	0h to 23h	**24decadehistoryofpopularmusic, sawtaylormac, 24decades, proofoflifenumber, marskado.**	0h to 23h	**doomocracy, pedroreyes, doomacracy, creativetime.**

Table 5.2: Top 10 events returned by SIGLER-Cov, MED, and GeoBurst, in NYC dataset, for the first 10k tweets (the day of 8 Oct. 2016). True positive events are market in bold while false negative events are not.

We believe that the results 1, 2, 7, and 9 of MED, and the result 4 of GeoBurst are not relevant. Indeed, they are defined with terms that do not correspond to any real life event (e.g., nyc, job, hiring, foodporn, etc.). Concretely, The terms ”nyc, newyork, love, manhattan, job, hiring, newyorkcity, photo, repost” are very frequent in New York dataset. Each of them appear at least 30 times in 90% of the days. The term ”saturday” is frequently used in Saturday (more than 30 times in 80% of cases). The terms ”food, foodie, yummy, eeeeeats” also appear in a large number of posts where people want to share their feeling about some food experience. Each of them is used at least 7 times in 50% of the days. The term ”ridetheferry” is used by people who pass by the NY Waterway Ferry. It occurs between 1 and 6 times in 22 different days.

5.6.3 Efficiency

To evaluate the scalability of the algorithms, we consider New York tweets, our largest dataset. Fig. 5.7 reports the runtime and the number of events obtained by SIGLER-Cov[9], MED, and GeoBurst when

[9]The runtime for SIGLER-Cov corresponds to the complete process including the post-processing step described in Section 5.5.3. This explains the slight variation of the runtime of SIGLER-Samp for different configurations.

dataset parameters are varying. However, MED is only reported for at most 10,000 tweets because of its scalability issues (in the figures at left).

MED, GeoBurst, and SIGLER-Cov discover a comparable number of events but MED performances raise scalability issues. Indeed, SIGLER-Cov outperforms MED with several orders of magnitude for all the configurations, especially when the number of tweets increases. Even if MED uses some indexing techniques, its computational complexity is quadratic in the number of tweets, and MED fails to handle large datasets. Although MED is able to process one day of tweets in our experiments, it is without considering the fact that the Twitter API gives access to less than 1% of the posted tweets. MED scale limitation is therefore a real issue on these data.

In Fig. 5.7 - right, we report the runtime and the number of discovered events with higher numbers of tweets (we consider the whole dataset). We observe that the execution time of SIGLER-Cov increases with the number of tweets, but there is no order of magnitude change (non-logarithmic scale). Although GeoBurst runtime also increases linearly, it is considerably higher than the one of SIGLER-Cov.

We also study the impact of number of vertices and time granularity on our methods. We show their behavior in Fig. 5.8 when these two dimensions are varying. The execution time of SIGLER-Cov increases when the number of vertices and time granularity increase. The execution time of SIGLER-Samp is controlled by a parameter and we can observe that its number of results tends to the one of SIGLER-Cov when the time budget increases.

To go further on evaluating the discovered events, Fig. 5.9 investigates the ability of SIGLER-Samp to capture similar events as SIGLER-Cov, that is to say it verifies that the computed event sample well covers all the events obtained with the exhaustive approach. To this end, we compare the events provided by SIGLER-Samp (denoted R_1) with the events discovered by SIGLER-Cov (denoted R_2) as follows. Using the Jaccard similarity measure of two patterns $P1 = (U_1, I_1)$ and $P_2 = (U_2, I_2)$:

$$J(P_1, P_2) = \frac{1}{3} \times \left(\frac{|U_1 \cap U_2|}{|U_1 \cup U_2|} + \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|} + \frac{|H_{P1} \cap H_{P2}|}{|H_{P1} \cup H_{P2}|} \right),$$

the similarity of R_1 and R_2 is defined as:

$$Sim(R_1, R_2) = \frac{\sum_{P_1 \in R_1} \max_{P_2 \in R_2} J(P_1, P_2)}{|R_1| + |R_2|} + \frac{\sum_{P_2 \in R_2} \max_{P_1 \in R_1} J(P_1, P_2)}{|R_1| + |R_2|}.$$

We executed SIGLER-Samp with different time budgets and post-processed the result R_2 by removing redundant patterns (using $\texttt{minCov} = 0.8$) and low quality ones (using $\delta = 40$). Fig. 5.9 reports the Sim values with respect to SIGLER-Samp time budget for 300K tweets (left) and the whole dataset (right). The runtimes of SIGLER-Cov for these two cases are respectively 95s and 173s. With a time budget fixed to 11% of the execution time of SIGLER-Cov (around 9s and 19s), SIGLER-Samp retrieves most of the high-quality patterns ($sim > 0.9$).

Finally, Fig. 5.10 evaluates the impact of the two main parameters, `minCov` and δ, on the results. `minCov` is a very intuitive parameter that eliminates a pattern if it is covered at least by `minCov`% of a pattern belonging to the solution. When `minCov=1`, only non maximal patterns are removed, and the more `minCov` decreases, the more disjoint the patterns. From Fig. 5.10 we can observe that this parameter

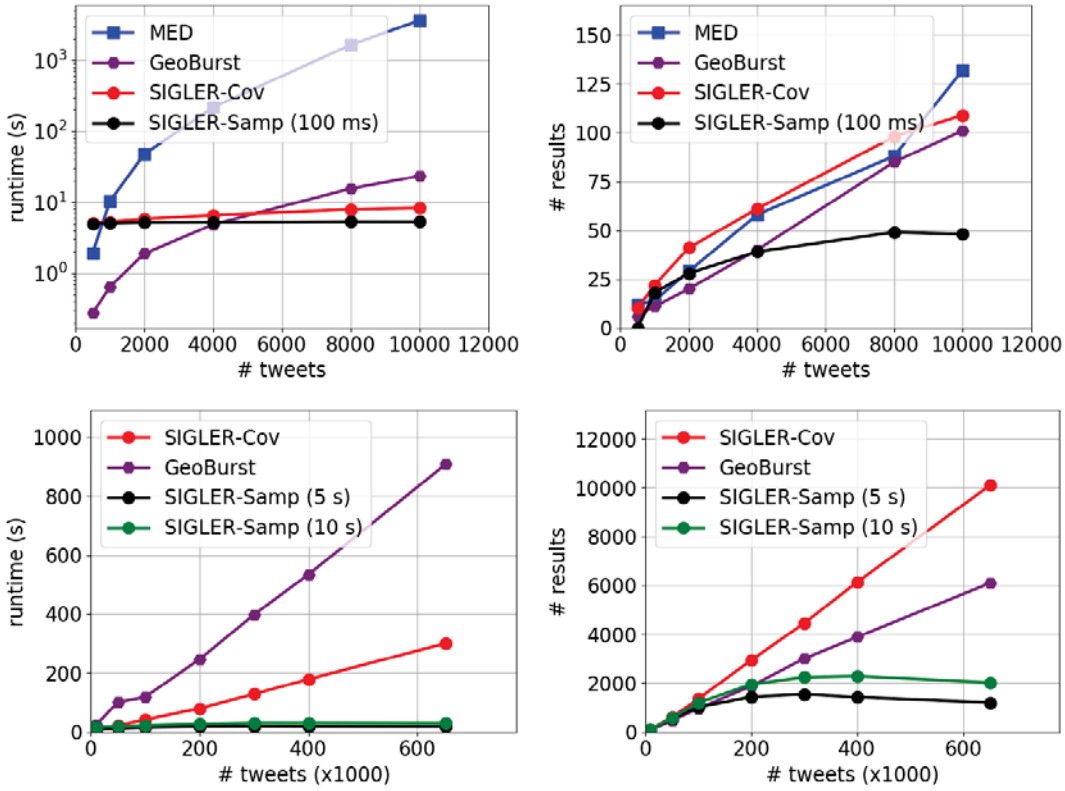


Figure 5.7: Runtime and number of events of SIGLER-Cov, SIGLER-Samp, MED and GeoBurst when varying the number of tweets (default values of SIGLERs: 2000 vertices, $\Delta t = 3h$, $\delta = 10$ and $\texttt{minCov} = 0.8$).

has also a major impact on the execution time. Indeed, this parameter is involved in the computation of the upper-bounds and when large, it drastically reduces the execution time. In our experiments, we set this parameter to 0.8 to remove highly redundant events while allowing some intersections.

The quality of an event is evaluated by the measure $\mathcal{M}$, also used to rank the patterns when presented to the user (see next subsections). The function of the parameter δ is to cut the tail of the pattern distribution in order to only keep those of high quality. So the larger δ, the smaller the number of patterns and the faster the execution.

For the following experiments, we fixed the value of δ according to the number of events that we wanted to present to the user. Thus, we fixed δ value so as to have around 800 events for NYC, which contains 3 months of tweets, and around 400 events for LA and London that contain 70 days of tweets. This led us to set $\delta = 40$ for NYC, and $\delta = 15$ for London and LA dataset.

5.6.4 User-driven discovery of geo-located events

Evaluation with real users. To evaluate the ability of SIGLER-Cov to take benefit of user feedback, we performed interactive event detection process with real users on real datasets. We used a crowdsourcing

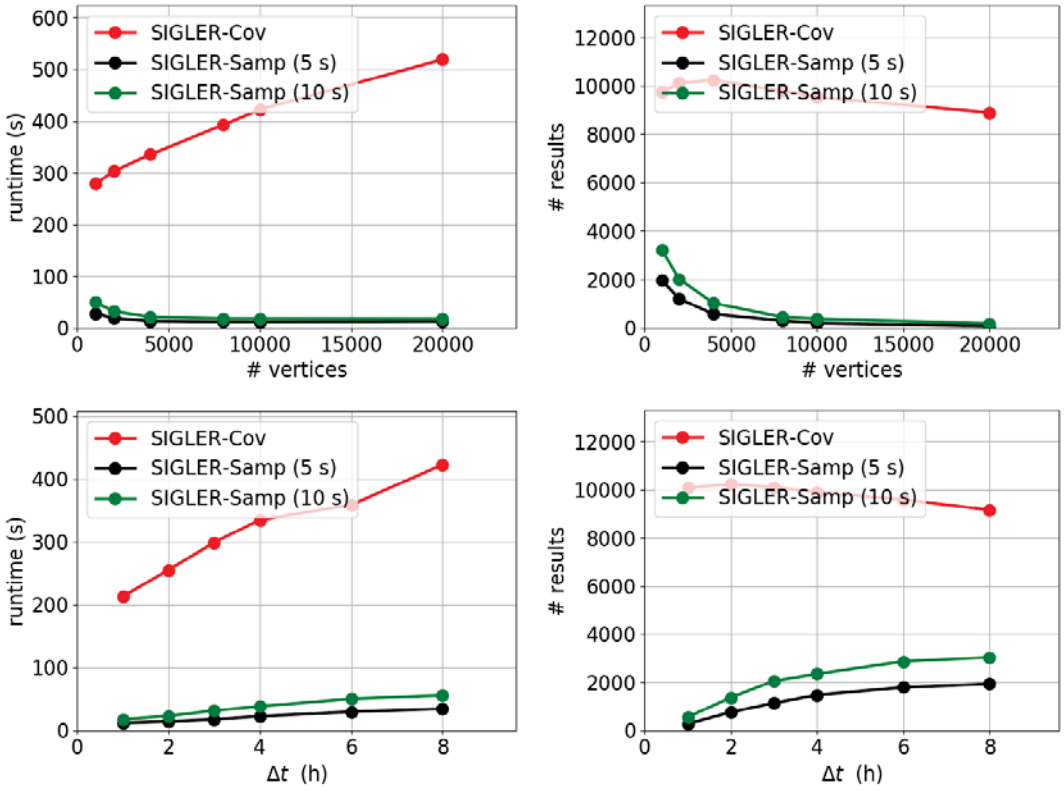


Figure 5.8: Number of events by SIGLER-Cov and SIGLER-Samp and runtime of SIGLER-Cov according to the number of vertices, and the time granularity when considering the whole dataset (default values: 652k tweets, 2000 vertices, $\Delta t = 3h$, $\delta = 10$ and $\texttt{minCov} = 0.8$).

platform – Figure Eight[10]– to hire people living in the country where the data is located. Indeed, our user feedback requires some expertise about the city and its events. To this end, we developed a graphical application[11] and deployed it on Figure Eight. For each user, the process consists in several iterations. At each of them, a batch containing the tweets emitted for 6 consecutive days is given as input to the algorithms (with a recovery of 3 days between 2 batches). Geolocated events are computed using either $\mathcal{M}$ (data-driven detection), or $\mathcal{M}_{ud}$ (user-driven detection). Then, the user is asked to mark the events that she likes. Finally, the batch index is incremented and the process iterates. Since evaluating with the overall datasets can be very long and exhausting for real users, we limited the number of batches to 10, that is approximately 33 days of data.

We used two different settings to compare the data-driven and the user-driven approaches. In the *paired sample* (also referred to as comparative evaluation), each of the 40 participants evaluated both approaches in a blind way, i.e. the two lists of events were randomly displayed to the user[12]. In the *unpaired sample* (also known as independent evaluation), 60 participants evaluated either the data or the

[10] `https://www.figure-eight.com`
[11] `134.214.104.134:6001` and `134.214.104.134:6002`.
[12] The paired sampled evaluation framework is available at: `134.214.104.134:6002`

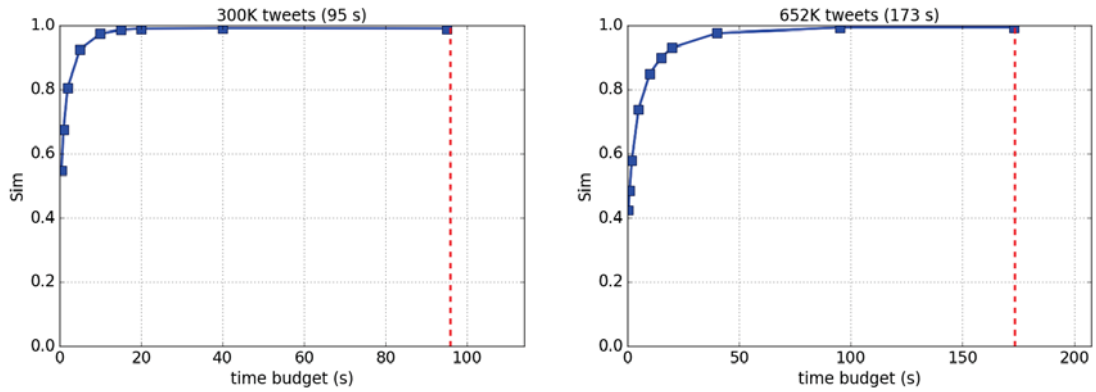


Figure 5.9: Similarity of the results of SIGLER-Cov with the ones of SIGLER-Samp with respect to SIGLER-Samp time budget.

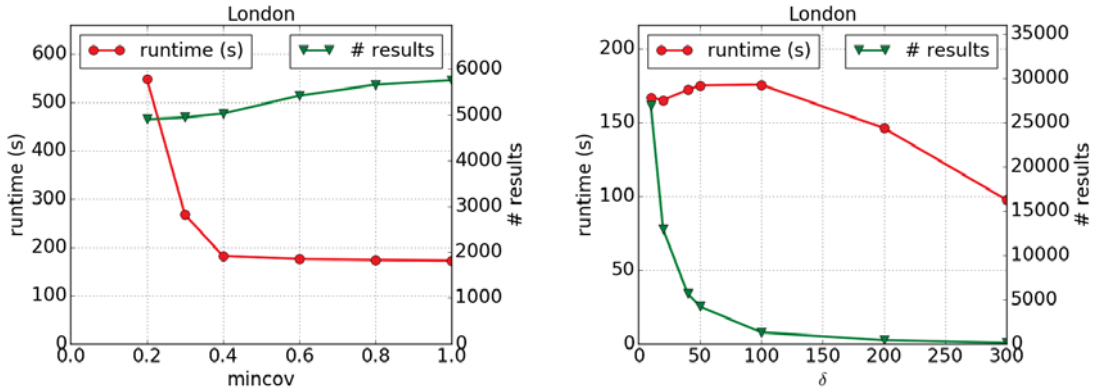


Figure 5.10: Runtime and number of discovered events by SIGLER-Cov according to `minCov` and δ (default value `minCov`=0.8, $\delta = 40$).

user-driven approach while not being aware of the type of method used[13].

Fig. 5.11 reports the results of this crowdsourcing-based evaluation. For the paired sample, the number of likes is greater or equal in the user-driven setting than in the data-driven one, while results are less obvious for the unpaired sample. The purpose of paired samples is to get better statistics by controlling for the effects of other "unwanted" variables. And so, as our sample sizes are quite small, results obtained on paired samples are probably the most reliable. In addition, the test of Wilcoxon [63] is applicable on the *paired sample*, while it is not on the *unpaired* ones. However, even for paired samples, the Wilcoxon test does not allow to reject the null hypothesis "the number of likes are similar", and the difference is not considered as significant. Considering the average ranks of liked events, we can observe that they are always better in the user-driven configuration than in the data-driven one. The difference in the values is considered as significant by both Wilcoxon and Nemenyi tests [63]. The Nemenyi test value

[13]The unpaired evaluation framework is available at: `134.214.104.134:6001`.

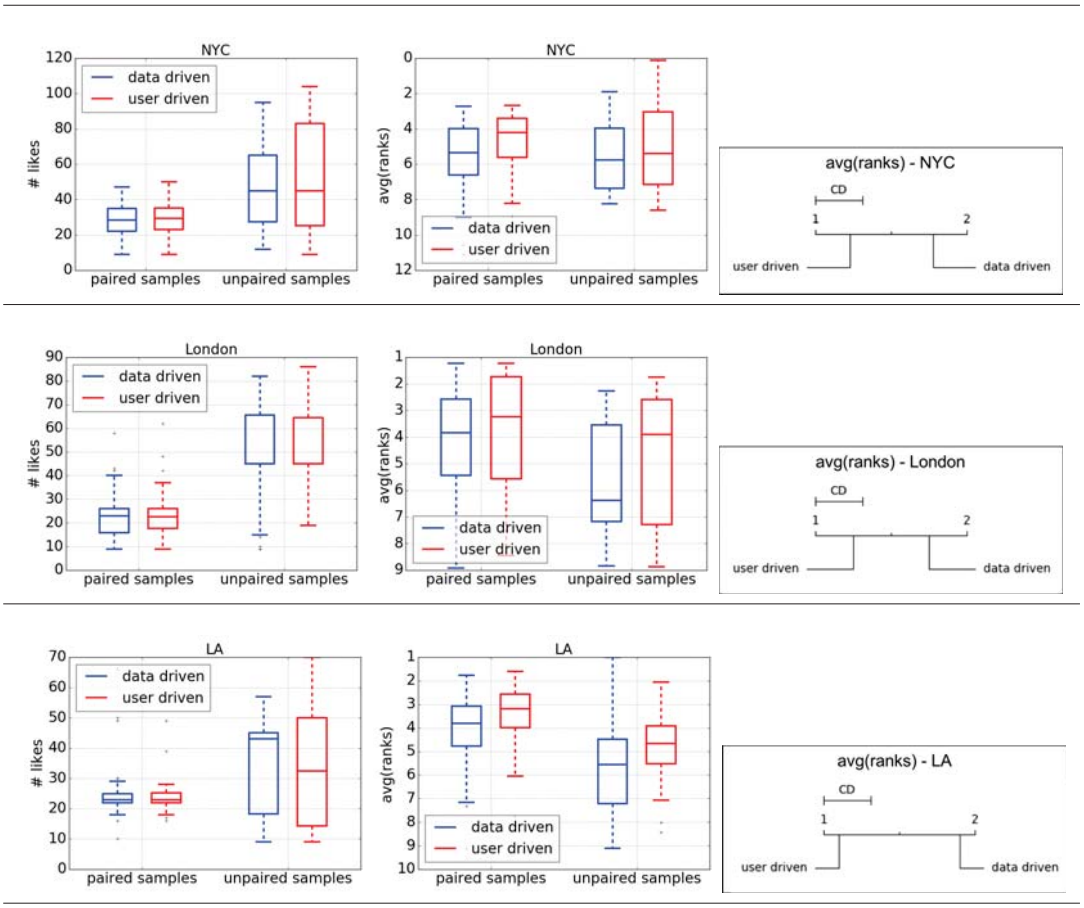


Figure 5.11: Ability to take user feedback into account with real users: number of likes (left), average ranks of liked events (center), Nemenyi tests on average ranks (right) for paired samples.

is shown in Fig. 5.11: when the rank difference on the graduated line is greater than the CD value, the rank difference is considered not due to chance.

Thus, this experience with real users leads to nuance the claim that the user-driven approach makes it possible to identify more interesting events than the data-driven one. However, it confirms that the identified events are of much better quality for the user-driven setting than for the data-driven one. Similar experiments with simulated users are reported in supplemental material.

Evaluation with synthetic users. we provide supplementary experiments of the user-driven approach. We simulated virtual users who, depending on their "own interest", tend to prefer events of a given type. We evaluate their "satisfaction" according to the number of events that the system presents to them that they are inclined to like. Thus, virtual users having topics and/or location preferences are simulated and the experiment consists in studying the number of liked events when considering or not user feedback during the event discovery process.

To that end, we first extracted events in the three real datasets[14] and retain those spanning at least over

[14]To obtain around 800 events on NYC, we used $\delta = 40$, and to obtain around 400 events on LA and London, we used

2 timestamps (the set $\mathcal{E}$). Then, we manually annotated the events. The tags used for the annotation are Topics $=$ {Business/Economics, Politics, Science/Technology, Art/Culture, Celebration, Music, Sport, Accident/Disaster}. Each event P can be annotated with several topics and the function $\text{Tag}(P,\tau) \rightarrow [0,1]$ expresses the importance of the topic for the event (with $\sum_{\tau \in \text{Topics}} \text{Tag}(P,\tau) = 1$). Some detected events did not match to any category and the obtained distributions are presented in Table 5.3.

dataset	#	Art/Culture	Music	Celebration	Sport	Politics	Business
NYC	800	97	89	212	87	88	1
LA	489	157	70	18	30	0	7
London	353	120	32	8	53	3	36

Table 5.3: Distribution of events according to the topics.

A virtual user u prefers a specific topic, or location, or both of them. The function $\mathcal{P}_u(P)$ captures the preferences of the user u for the event P. It is defined according to 3 cases:

- If τ is the preferred topic of u, $\mathcal{P}_u(P) = \text{Tag}(P,\tau)$
- If ℓ is the preferred location of u, we consider that u is interested in events at a distance from ℓ at most equal to L and $\mathcal{P}_u(P) = \max(\frac{L-\text{dist}(\ell,P)}{L}, 0)$. Based on the surface area of the cities, we choose $L = 5km$ for New York, and $L = 10km$ for Los Angeles and London.
- If u has both topic and location interests, $\mathcal{P}_u(P) = \frac{\text{Tag}(P,\tau)+\max(\frac{L-\text{dist}(\ell,P)}{L},0)}{2}$

Topics with fewer than 20 events were discarded. For the preferred locations, we consider several well-known places for each city[15]. Finally, to be able to automatically annotating computed events on these datasets, we used the Tag function to annotate hashtags and locations (for $x = h, v$, $\text{Tag}'(x,\tau) = \sum_{P \in \mathcal{E}\ s.\ t.\ x \in P} \text{Tag}(P,\tau)$) and then use them to automatically annotate events ($\text{Tag}^\star(P,\tau) = \sum_{x \in P} \text{Tag}'(x,\tau)$).

To evaluate how SIGLER-Cov and SIGLER-Samp effectively discover user-driven events, we simulate the same interactive process that we did with the real users, but with virtual users this time.

Fig. 5.12 presents results of these experiments when simulating between 19 and 29 virtual users depending on the number of considered locations and topics on each dataset. For each dataset, we show boxplots of the average number of likes using SIGLER-Cov and SIGLER-Samp in the data and user-driven settings. We can observe that (1) the average number of likes in the user-driven setting is always greater than the one in data-driven configuration. This difference is considered as significant by the Wilcoxon and the Nemeny post-hoc [63] tests (the later is shown on Fig. 5.12. (2) results obtained by SIGLER-Samp are below those obtained by SIGLER-Cov, and the difference is significant according to the Wilcoxon test, except for the NYC dataset, where the number of likes is considered to be similar.

$\delta = 15$. On all datasets we set $\texttt{minCov} = 0.8$.

[15] **NYC**: Barclays Center, Javits Center, Madison Square Garden and Metlife Stadium; **LA**: City Hall, DisneyLand, Museum of Art and Rose Bowl Stadium; **London**: City of London, Royal Albert Hall, Soho Theatre and Wembley Stadium.

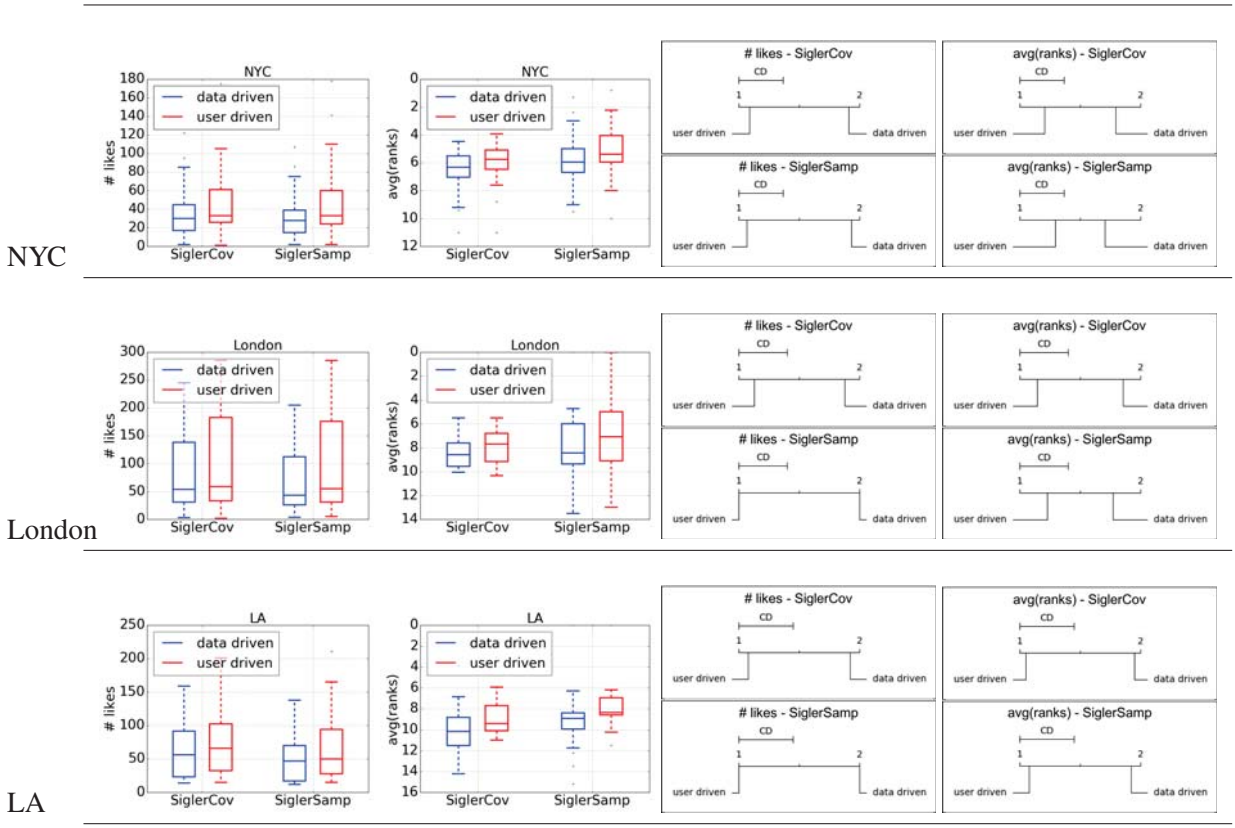


Figure 5.12: Virtual user-driven geo-spatial event detection with SIGLER-Cov and SIGLER-Samp. Number of likes (1st column) and average ranks of events liked by both data and user-driven settings (2nd column), Nemenyi tests on number of likes and average ranks (3rd and 4th columns).

Indeed, SIGLER-Cov is more exhaustive and finds more events than SIGLER-Samp, which explains this point.

Notice that in this simulation between 300 and 800 events are presented to the virtual-users who, on the contrary to human users, have the ability to evaluate all of them. As already mentioned above, the order in which the events are presented to the users is essential. In order to evaluate this point, Fig. 5.12 presents the average ranks of events liked by both data and user-driven settings. Clearly the setting for which the liked events are ranked first is advantageous in real situations. We can observe that the average rank is always lower in the user-driven setting than in the data-driven one. This difference is considered significant by the Wilcoxon test on all datasets. This is also confirmed, in a more visual way, by the Nemenyi tests [63] displayed on the figure. When comparing the average ranks obtained by SIGLER-Cov and SIGLER-Samp, it appears that the later obtains significantly better ranks (smaller) for liked events that the former. Thus SIGLER-Samp identifies fewer events, but that are of high quality for users.

This simulation with virtual users allows us to conclude that the user-driven setting makes it possible to identify more relevant events than the data-driven one, whether in terms of quantity and quality. Besides, SIGLER-Samp identifies fewer geolocated events than SIGLER-Cov, but they are of better

quality.

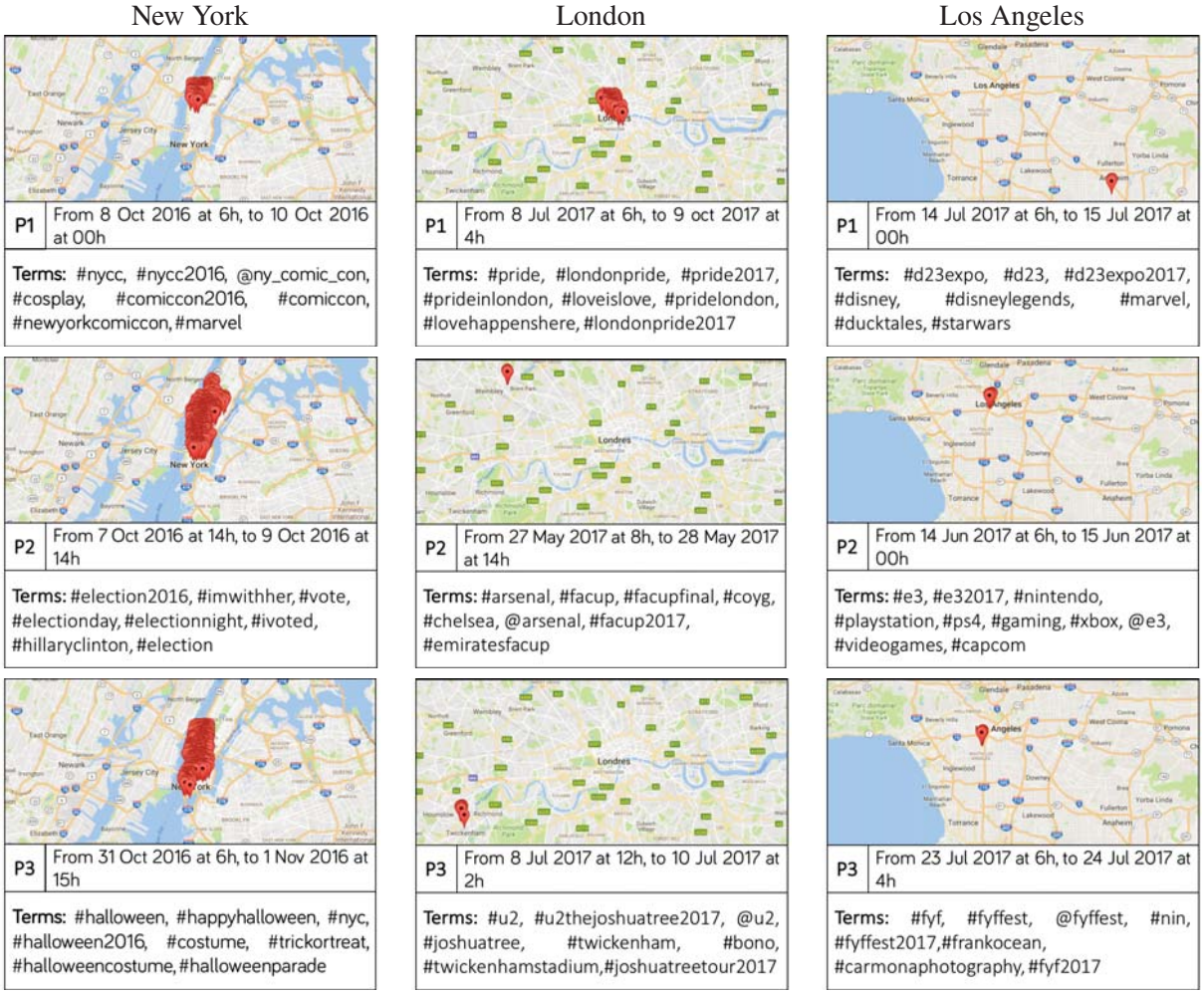


Figure 5.13: Top 3 events detected in New York (left), London (center), and Los Angeles (right).

5.6.5 Illustrative results

Finally, we show some examples of events detected by our approach. Fig. 5.13 reports the top 3 events detected in New York, London, and Los Angeles datasets. Each event is described by the locations, time interval and top 8 most frequent terms of its related tweets. The first event in New York is the Comic Con[16] , which is an annual convention mainly dedicated to comics. It was organized in Jacob K. Javits Convention Center, the location associated to the related tweets. The two other events correspond to USA presidential elections, and the celebration of Halloween. In London, the first event is a Pride Parade organized in July 8th, 2017. The remaining are two geolocated events corresponding to a soccer event

[16] `https://goo.gl/BR7kgp`

and a concert of U2. In Los Angeles, the top 3 events are respectively: the Official Disney Fan Club, E3[17] (a video game related event), and the FYF Fest[18] (a music festival).

5.7 Conclusion

In this chapter, we have introduced a new model for mining local patterns in vertex-attributed graphs, that is able to incorporate user preferences through an interactive process. This model is exploited to define the novel problem of user-driven geolocated event discovery in social media. We have designed two different algorithms to efficiently and effectively discover local patterns (i.e., events) based on the input data and user feedback. In the experiments, we have compared our approach against the state of the art event detection methods in a data-driven setting, i.e., without integrating user feedbacks. These experiments have demonstrated that the runtime of our approach outperforms state of the art methods by a factor of two to several orders of magnitude. Furthermore, the proposed approach is more robust to noise. We have also shown the ability of our method to discover local patterns that are of interest for the user based on her feedback with crowdsourcing-based experiments.

While user-preferences have been integrated in the interestingness model of this approach, her background knowledge has been ignored. In contrast, Chapter 4 proposed a method that incorporates user's background knowledge while ignoring her preferences. Considering both these factors in the same interestingness model is a more challenging problem which would allow to enhance the task of attributed subgraph mining. It requires to derive a model that is able to identify surprising (informative) patterns that belong to the topic of user's interest. Another interesting direction is to learn an explicit model of user interests and provide active learning based heuristics to foster the interactive process. The approach proposed in this chapter is a first attempt to learn user preferences in the attributed subgraph mining and event detection tasks. More sophisticated models can be studied.

[17] https://goo.gl/pio4dS
[18] https://goo.gl/k1yWmK

CHAPTER 6

Integrating Attribute Hierarchy into Subjective Interestingness

Contents

6.1 Introduction

In many datasets, attributes that are used to describe objects (or vertices) are organized by a hierarchical relationship. In Chapters 3 and 4, we have performed experiments on data collected from Foursquare and modelled as attributed graph. Foursquare provides a hierarchy[1] of venue types that we have used as attributes for the vertices. A subset of this hierarchy is shown in Fig. 6.1. This hierarchy gives a great opportunity to improve the quality of patterns returned to the user. For example, once a user knows that a city has many food venues, she will increase her expectations about the number of Portuguese restaurants, and vice versa. Also, this hierarchy means that there cannot be more Portuguese restaurants than food venues. In other terms, information about one attribute can affect one's expectation about another. Incorporating the hierarchy into the background model may avoid presenting redundant patterns. We address this challenge and we propose a subjective interestingness model that integrates dependencies implied by a hierarchical relation between attributes. To simplify the problem, we limit ourselves to the hierarchy structure of attributes that describe a dataset as a whole, and we study it independently from any graph structure.

Hierarchies have been extensively investigated in the literature for decades. In [14], the authors explained the high and promissing utility of ontologies in different steps of the KDD process, and proposed an association mining tool that benefits from ontologies in different stages of the mining process (data understanding, task design, etc.). The generic problem of Semantic Data Mining has been defined in [144, 209]: given a set of objects annotated with ontology terms, the goal is to find hypothesis, expressed by domain ontology terms, explaining the given empirical data. Specifically in [131, 209, 210], the Semantic Subgroup Discovery problem is studied: given a dataset where each object is annotated with ontology terms and belongs to a specific class, the goal is to find a conjunction of ontology terms (a conjunctive rule) that corresponds to a set of object discriminating a specific class. A similar method has been proposed in [14] to take benefit from ontology structures in order to optimize and reduce the redundancy in the search of conjunctive rules that satisfy some user specified constraints. In [21], this problem of rule learning is tackled based on ILP (Inductive Logic Programming), and ontology structures are integrated by providing additional clauses to the ILP solver. Most of these approaches have only considered hierarchies to restrain the pattern syntax or to take advantage of their subsumption power in order to optimize the exploration of the search space. Surprisingly, there has been little discussion about the use of the hierarchical relationships for reasoning in the knowledge discovery task. In our work, we explicitly incorporate the hierarchy into the background model used to quantify pattern interestingness. We aim to account for the inference that the user would make about values of these hierarchical attributes when information is provided to her about some specific subset of them.

In this chapter, we use the term "concept" to refer to the studied dataset (e.g., a city). A concept consists of a set of positive integer-valued attributes that are organized in a hierarchy. In Fig. 6.1, the concept is a city, and "Food" is an attribute whose value is 800. We attempt to characterize such concepts

[1] `https://developer.foursquare.com/docs/resources/categories`

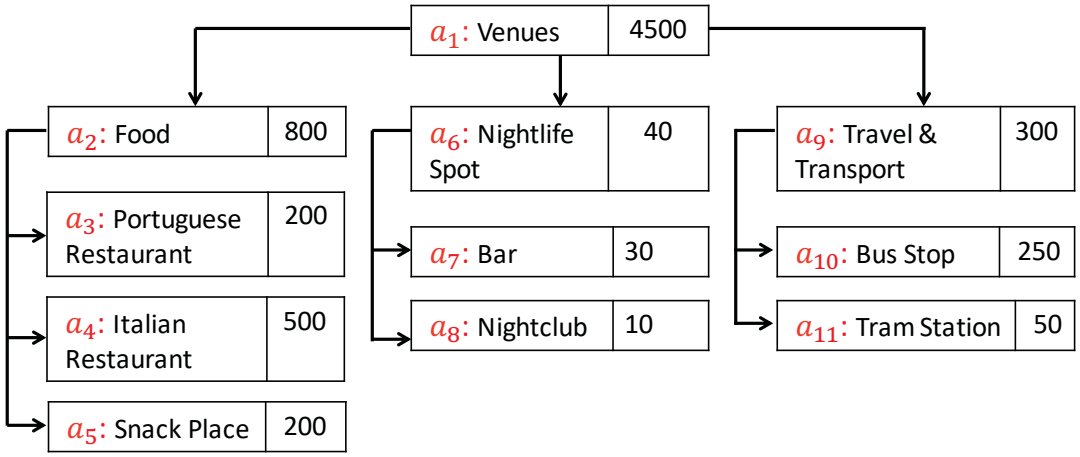


Figure 6.1: Example of hierarchy with counts (counts of venues in a city).

in terms of so-called *contrastive antichains*: particular kinds of subsets of incomparable attributes and their values. We address the question of when a contrastive antichain is interesting, in the sense that it concisely describes the unique aspects of the concept, and this while duly taking into account the known attribute dependencies implied by the hierarchy. Attributes belonging to the contrastive antichain need to have distinctive values. Hereby, distinctiveness is assessed with respect to some reference, e.g. the total sum of all concepts, or one concept in particular. The identified contrastive antichains pinpoint non-redundant concepts of the hierarchy that are at the same time compact and unexpected. In Fig. 6.1, the set $\{Food, Bar\}$ is an antichain, because none of these two attributes is a predecessor of the other one (they are not comparable). This antichain informs the user about the number of Food and Bar venues in the studied dataset. This antichain is informative if the number of Food and Bar venues are different from the expected values by the user.

This chapter is organized as follows. Section 6.2 introduces and formalizes the problem of mining *contrastive antichains* in hierarchically organized sets of attributes. Section 6.3 formalizes the *interestingness* of a contrastive antichain with respect to prior beliefs about the values of these attributes. These prior beliefs can be derived from a concept with which the concept under investigation needs to be contrasted, or from an aggregate count over all concepts. Section 6.3.3 in particular shows how the measure of interestingness can take into account the knowledge of previously found contrastive antichains, such that *iteratively mined* contrastive antichains can be made non-redundant. Section 6.4 presents an *algorithm* to efficiently mine the most contrastive antichains. Section 6.5 empirically evaluates the potential usefulness of contrastive antichains as a tool to gain insight into such data, and evaluate the effectiveness and scalability of the algorithm for mining them.

6.2 Contrastive antichains

Here we define the problem setting, along the way introducing the necessary concepts and notation. First we define the kind of data considered, and then the kinds of pattern we are interested in finding in such data.

6.2.1 The data: concepts described as sets of hierarchically related attributes

As discussed in the introduction, we consider the relatively common situation where concepts of interest are defined in terms of a set of positive integer-valued attributes, whereby these attributes are organized in a hierarchy. Indeed, these attributes values often represent a count of something in relation to that concept, e.g. the number of food venues (the attribute), in a particular city (the concept of interest). A particular concern in this work is the fact that these attributes are often related in a hierarchy, defined as follows.

Definition 6.1 (Hierarchy). A hierarchy (or a tree) $\mathcal{H}$ is defined as a tuple $\mathcal{H} = (A, \leq, a_1)$ where:

- $A = \{a_1, ..., a_n\}$ is a set of attributes,
- $\leq$ is a partial order relation defined over this set,
- $\forall a \in A : a_1 \leq a$ (the attribute a_1 is called the root of $\mathcal{H}$), and
- there is only one path from a_1 to any other attribute:

$$\forall a_i, a_j, a_k \in A : a_i \leq a_k \wedge a_j \leq a_k \Longrightarrow a_i \leq a_j \vee a_j \leq a_i.$$

In Fig. 6.1, the following relations hold: $a_1 \leq a_2$, $a_1 \leq a_3$, and $a_2 \leq a_3$. If $a_i \leq a_j$, then a_i is a predecessor of a_j, but a_i is not necessarily the direct parent of a_i. Note that a hierarchy is a special case of a partially ordered set. We assume such a hierarchy imposes certain constraints on the values these attributes may have. Specifically, we assume that the value of an attribute is never strictly larger than the value of a hierarchically smaller attribute in the partial order.

For convenience, we introduce the following operations over a hierarchy.

Definition 6.2 (Basic operations). Given $S \subseteq A$:

- Precedessors operator $\Uparrow$, and successors operator $\Downarrow$ as:

$$\Uparrow S = \{a \in A \mid \exists a' \in S : a \leq a'\},$$
$$\Downarrow S = \{a \in A \mid \exists a' \in S : a' \leq a\},$$

- Strict predecessor relation: $a_i < a_j \Leftrightarrow a_i \leq a_j \wedge a_i \neq a_j$,
- The direct successor relation $\prec$ as: $a_i \prec a_j \Longleftrightarrow \Downarrow a_i \cap \Uparrow a_j = \{a_i, a_j\}$. Also, if $a_i \prec a_j$, we use the notation $\pi_i = j$ to refer to the index of the direct parent of a_i ($a_j = a_{\pi_i}$),

- Direct predecessor operator $\uparrow$, and direct successors operator $\downarrow$ as:

$$\uparrow S = \{a \in A \mid \exists a' \in S : a \prec a'\},$$
$$\downarrow S = \{a \in A \mid \exists a' \in S : a' \prec a\},$$

- the leaves $L(\mathcal{H}) = \{a \in A \mid \nexists a' : a \prec a'\}$,

- The mimimum of S: $\min(S) = \{a \in S \mid \nexists a' \in S : (a' < a)\}$.

The value of each attribute $a_i \in A$ is denoted using the variable x_i, and when a particular empirical value is meant it will be denoted as $\hat{x}_i$. Note that this means that $\forall i \in [\![2, n]\!] : x_i \leq x_{\pi_i}$ as well as $\hat{x}_i \leq \hat{x}_{\pi_i}$.

6.2.2 Contrastive antichains as patterns

We aim to inform the analyst about the values of a subset of the attributes of a concept. As formalized in Section 6.3, it is our goal to ensure that the patterns of this type are as informative to the data analyst as possible, taking into account the fact that the data analyst has certain prior beliefs about values of these attributes, e.g. based on one or a set of other well-understood concepts with the same sets of attributes. As information on an attribute directly affects one's expectations about comparable (i.e. hierarchically related) attributes (see Section 6.3 where we make this rigorous), we made the choice of including into the same pattern only non-comparable attributes—i.e. no attribute is a predecessor or successor of another, and the set of attributes forms an *antichain* of the hierarchy. This ensures in a constructive manner that the information provided by the different attributes in the pattern is not too redundant, and ensures that the patterns provide information about a larger part of the hierarchy.

For each attribute $a_i \in P$, a contrastive antichain pattern informs the data analyst about the value $\hat{x}_i$. The question arises whether it is of interest to inform the analyst about its precise value. We argue that in many practical cases the precise value gives no more insight than an order of magnitude indication (although probably more detailed than orders of 10). Thus, instead of informing the analyst about the precise value, in this work we consider the case where the pattern describes just the scale of the count: $\lfloor \log(\hat{x}_i) \rfloor$.

A contrastive antichain pattern can thus be formally defined as:

Definition 6.3 (Contrastive antichains). Given a concept and the value $\hat{x}_i$ for each of a set of attributes $a_i \in A$ which are organized in a hierarchy $\mathcal{H} = (A, \leq, a_1)$. A contrastive antichain pattern P is specified by a subset of the attributes ($P \subseteq A$) which forms an antichain w.r.t. $\leq$, i.e., $\forall a_i, a_j \in P$: $a_i \leq a_j \implies a_i = a_j$, along with the integers $\lfloor \log(\hat{x}_i) \rfloor$ describing the scale of the values of these attributes.

6.3 The interestingness of a contrastive antichain

For a hierarchy $\mathcal{H} = (A, \leq, a_1)$, we aim to represent the prior beliefs of the user about the attributes $a_i \in A$. This will be represented by a probability distribution Pr for the set of random variables X_i. We call this distribution the *background distribution*. In practice, the prior beliefs are derived from one or a number of example concepts and the values of their attributes, as explained next.

6.3.1 The background distribution

In this study, we consider that the analyst has some reference that will determine her prior expectations about the attributes for the concept under investigation. For instance, if we consider cities as concepts, a Londoner who would like to gain an understanding of the city of Amsterdam in terms of its attributes values, will typically have prior beliefs determined by the values of those attributes in London. More specifically, we assume that the analyst has an expectation of the value x_i of each attribute $a_i \in A$ that is equal to $\bar{x}_i$, determined by the values of the respective attribute in the reference concept. In addition, we assume that the analyst has specific expectations about the value of each attribute (except for a_1) relative to its parent attribute's value. This can be formalized using the following two types of constraints:

- **Expectations**: the expectation of each random variable X_i is $\bar{x}_i$:

$$\forall i \in [\![1, n]\!] : \sum_{x_i} \Pr(x_i) \cdot x_i = \bar{x}_i, \tag{6.1}$$

- **Conditional expectations**: The expectation of the ratio of an attribute's value over the parent's attribute's value, *conditional on* that parent attribute's value, is equal to the observed ratio of these values:

$$\forall i \in [\![2, n]\!] : \sum_{x_i} \Pr(x_i | x_{\pi_i}) \cdot \frac{x_i}{x_{\pi_i}} = \frac{\bar{x}_i}{\bar{x}_{\pi_i}}. \tag{6.2}$$

For $i > 1$, the second type of constraint is arguably a more accurate encoding of the analyst's prior expectations, as it explicitly encodes dependencies of attributes conditioned on parent attributes. For instance, for cities as concepts and the venues hierarchy, the analyst may expect that 75% of nightlife spots are bars, and knowing the number of nightlife spots will arguably affect the expectation of the number of bars. Moreover, the following property shows that the second constraint essentially subsumes the former, i.e. it is strictly stronger than the former.

Property 6.1. If the two following sets of conditions hold:

1. **Expectations for only the root**: $\sum_{x_1} \Pr(x_1) \cdot x_1 = \bar{x}_1$,
2. **Conditional expectations for other nodes**: $\forall i \in [\![2, n]\!] : \sum_{x_i} \Pr(x_i | x_{\pi_i}) \cdot \frac{x_i}{x_{\pi_i}} = \frac{\bar{x}_i}{\bar{x}_{\pi_i}}$,

Then it follows that $\forall i > 1, \sum_{x_i} \Pr(x_i) \cdot x_i = \bar{x}_i$.

Proof. We consider that $\sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot x_{\pi_i} = \bar{x}_{\pi_i}$ and we prove that $\sum_{x_i} \Pr(x_i) \cdot x_i = \bar{x}_i$. The complete proof can be recursively established by starting from the root ($x_{\pi_i} = x_1$) for which we already know that $\sum_{x_1} \Pr(x_1) \cdot x_1 = \bar{x}_1$.

$$\begin{aligned}
\sum_{x_i} \Pr(x_i) \cdot x_i &= \sum_{x_i} \left(\sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot \Pr(x_i|x_{\pi_i}) \right) \cdot x_i, \\
&= \sum_{x_i} \sum_{x_{\pi_i}} \left(\Pr(x_{\pi_i}) \cdot \Pr(x_i|x_{\pi_i}) \cdot x_i \right), \\
&= \sum_{x_{\pi_i}} \sum_{x_i} \left(\Pr(x_{\pi_i}) \cdot \Pr(x_i|x_{\pi_i}) \cdot x_i \right), \\
&= \sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot \sum_{x_i} \left(\Pr(x_i|x_{\pi_i}) \cdot x_i \right), \\
&= \sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot \frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot x_{\pi_i}, \\
&= \frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot \sum_{x_{\pi_i}} \Pr(x_{\pi_i}) \cdot x_{\pi_i}, \\
&= \frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot \bar{x}_{\pi_i}, \\
&= \bar{x}_i.
\end{aligned}$$

This means that it suffices to consider the expectation constraint for only the root, in addition to conditional expectation constraints for the other nodes.

Thus we assume that the joint distribution can be written as:

$$\Pr(x_1, x_2, \ldots, x_n) = \Pr(x_1) \prod_{i \in [\![2,n]\!]} \Pr(x_i, x_{\pi_i}). \tag{6.3}$$

The constraints on the probability for x_1 and the conditional probabilities of x_i conditioned on x_{π_i} do not uniquely define the joint probability distribution of all attribute values. As proposed in [60], we use distributions that maximize the entropy, as any other distribution effectively makes additional assumptions that reduce the entropy. This means that these marginal distributions are found as the solutions of the following optimization problem:

- For $i = 1$ (x_i is the root):

$$\begin{aligned}
\max_{\Pr(x_i)} \; & -\sum_{x_i} \Pr(x_i) \log \Pr(x_i), \\
\text{s.t.} \; & \sum_{x_i} \Pr(x_i) x_i = \bar{x}_i, \\
& \sum_{x_i} \Pr(x_i) = 1.
\end{aligned}$$

- For $i > 1$, the distribution $\Pr(x_i|x_{\pi_i})$ needs to maximize the entropy conditional on x_{π_i}. Here, we must however recognize that the value x_i is derived from its parent attribute x_{π_i} by determining which of the elements counted as part of x_{π_i} also contribute to x_i. I.e., the prior expectation is about the tendency of any element contributing to x_{π_i} to also contribute to x_i. In terms of such decisions for individual elements, there are many ways to arrive at the same count, namely $Q(x_i, x_{\pi_i}) = \binom{x_{\pi_i}}{x_i}$ ways. Thus, we should use $Q(x_i, x_{\pi_i})$, the number of ways to realize this attribute value, as a base measure, or equivalently, maximize the Kullback-Leibler divergence between the conditional probability $\Pr(x_i|x_{\pi_i})$ and $Q(x_i, x_{\pi_i})$:

$$\begin{aligned} \max_{\Pr(x_i|x_{\pi_i})} & -\sum_{x_i} \Pr(x_i|x_{\pi_i}) \log \frac{\Pr(x_i|x_{\pi_i})}{Q(x_i, x_{\pi_i})}, \\ \text{s.t.} & \sum_{x_i} \Pr(x_i|x_{\pi_i}) x_i = \frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot x_{\pi_i}, \\ & \sum_{x_i} \Pr(x_i|x_{\pi_i}) = 1. \end{aligned}$$

The solution to the first problem is a *geometric distribution* [60] having an expectation equal to $\bar{x}_1$:

$$\Pr(x_1) = \left(1 - \frac{1}{1+\bar{x}_1}\right)^{x_1} \cdot \frac{1}{1+\bar{x}_1} = (1-p_1)^{x_1} \cdot p_1,$$

with $p_1 = \frac{1}{1+\bar{x}_1}$.

The solution to the second problem for the other random variables x_i ($i > 1$), is a *binomial distribution* with an average $\frac{\bar{x}_i}{\bar{x}_{\pi_i}} \cdot x_{\pi_i}$ [105]:

$$\Pr(x_i|x_{\pi_i}) = \binom{x_{\pi_i}}{x_i} \cdot \left(\frac{\bar{x}_i}{\bar{x}_{\pi_i}}\right)^{x_i} \cdot \left(1 - \frac{\bar{x}_i}{\bar{x}_{\pi_i}}\right)^{x_{\pi_i} - x_i} = \binom{x_{\pi_i}}{x_i} \cdot b_i^{x_i} \cdot (1-b_i)^{x_{\pi_i} - x_i},$$

where $b_i = \frac{\bar{x}_i}{\bar{x}_{\pi_i}}$ is the binomial parameter.

Property 6.2. The marginal distribution for each random variable x_i is geometric, and it is:

$$\Pr(x_i) = \left(1 - \frac{1}{1+\bar{x}_i}\right)^{x_i} \cdot \frac{1}{1+\bar{x}_i}.$$

Proof. For the root x_1, this is already the case. We then prove this property by recursion, we consider that x_{π_i} follows a geometric distribution with parameter $p_{\pi_i} = \frac{1}{1+\bar{x}_{\pi_i}}$ and we prove that x_i also follows a geometric distribution with a parameter $p_i = \frac{1}{1+\bar{x}_i}$:

$$\Pr(x_i) = \sum_{x_{\pi_i}=x_i}^{+\infty} \Pr(x_{\pi_i}) \cdot \Pr(x_i|x_{\pi_i}), \tag{6.4}$$

$$= \sum_{x_{\pi_i}=x_i}^{+\infty} (1-p_{\pi_i})^{x_{\pi_i}} \cdot p_{\pi_i} \cdot \binom{x_{\pi_i}}{x_i} \cdot b_i^{x_i} \cdot (1-b_i)^{x_{\pi_i}-x_i}, \tag{6.5}$$

$$= \frac{p_{\pi_i} \cdot b_i^{x_i}}{(1-b_i)^{x_i}} \cdot \sum_{x_{\pi_i}=x_i}^{+\infty} \binom{x_{\pi_i}}{x_i} ((1-p_{\pi_i})(1-b_i))^{x_{\pi_i}}, \tag{6.6}$$

$$= \frac{p_{\pi_i} \cdot b_i^{x_i}}{(1-b_i)^{x_i}} \cdot \sum_{k=0}^{+\infty} \binom{k+x_i}{x_i} ((1-p_{\pi_i})(1-b_i))^{k+x_i}, \tag{6.7}$$

$$= \frac{p_{\pi_i} \cdot b_i^{x_i}}{(1-b_i)^{x_i}} \cdot ((1-p_{\pi_i})(1-b_i))^{x_i} \cdot \sum_{k=0}^{+\infty} \binom{k+x_i}{x_i} ((1-p_{\pi_i})(1-b_i))^{k}, \tag{6.8}$$

$$= p_{\pi_i} \cdot b_i^{x_i} \cdot (1-p_{\pi_i})^{x_i} \cdot \sum_{k=0}^{+\infty} \binom{k+x_i}{k} ((1-p_{\pi_i})(1-b_i))^{k}, \tag{6.9}$$

The negative binomial infinite series can be simplified:

$$\sum_{k=0}^{+\infty} \binom{k+x_i}{k} ((1-p_{\pi_i})(1-b_i))^{k} = (1-(1-p_{\pi_i})(1-b_i))^{-1-x_i},$$
$$= (b_i + p_{\pi_i} - b_i \cdot p_{\pi_i})^{-1-x_i},$$

then:

$$\Pr(x_i) = p_{\pi_i} \cdot b_i^{x_i} \cdot (1-p_{\pi_i})^{x_i} \cdot (b_i + p_{\pi_i} - b_i \cdot p_{\pi_i})^{-1-x_i}, \tag{6.10}$$

$$= \frac{p_{\pi_i}}{b_i + p_{\pi_i} - b_i \cdot p_{\pi_i}} \cdot (1 - \frac{p_{\pi_i}}{b_i + p_{\pi_i} - b_i \cdot p_{\pi_i}})^{x_i}. \tag{6.11}$$

After substituting $p_{\pi_i} = \frac{1}{1+\bar{x}_{\pi_i}}$ and $b_i = \frac{\bar{x}_i}{\bar{x}_{\pi_i}}$, we find that $\frac{p_{\pi_i}}{b_i + p_{\pi_i} - b_i \cdot p_{\pi_i}} = \frac{1}{1+\bar{x}_i}$. This means that:

$$\Pr(x_i) = (1 - \frac{1}{1+\bar{x}_i})^{x_i} \cdot \frac{1}{1+\bar{x}_i}.$$

This concludes the proof.

6.3.2 The interestingness of an antichain

We use the framework of subjective interestingness SI proposed in [60], that defines $SI(P)$ as the ratio between the informativeness of a pattern and its description length (cost of communication to the user):

$$SI(P) = \frac{IC(P)}{DL(P)}.$$

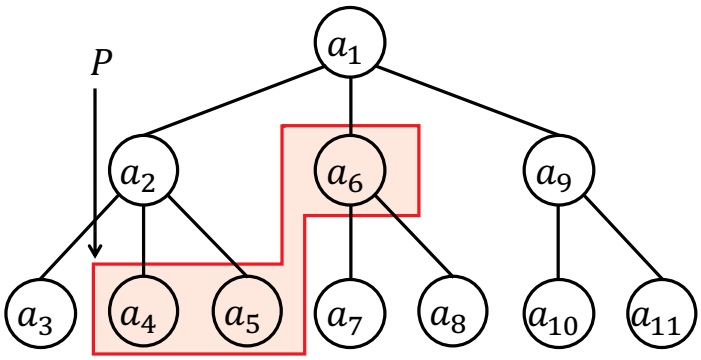


Figure 6.2: Example of an antichain pattern $P = \{a_4, a_5, a_6\}$

Information content IC: The information carried by a pattern is quantified by the information content [60], a quantity also known as the self-information or surprisal [58]. In particular, it is equal to the reduction of uncertainty about the data caused by the knowledge of the pattern, and is defined as follows:

$$IC(P) = -\log(\Pr(P)).$$

As previously discussed, the information content of a pattern is shared with the user by transmitting the scales of the counts appearing in the pattern, instead of the exact values, which can be overwhelming and difficult to remember. Let us define the random variable $Y_i = \lfloor \log(X_i) \rfloor$, whose true value is $\hat{y}_i = \lfloor \log(\hat{x}_i) \rfloor$. The probability associated to a pattern P is thus:

$$\begin{aligned}\Pr(P) &= \prod_{e_i \in P} \Pr(\hat{y}_i) = \prod_{e_i \in P} \Pr(2^{\hat{y}_i} \leq x_i < 2^{\hat{y}_i+1}), \\ &= \prod_{e_i \in P} \left((1-p_i)^{2^{\hat{y}_i}} - (1-p_i)^{2^{\hat{y}_i+1}} \right).\end{aligned}$$

Description length DL: The description length measures the complexity of communicating a pattern P to the user. In general, the closer an attribute $a \in A$ is to the root, the more likely the user is to know it. For example in Foursquare hierarchy, it is simpler to communicate the node "Asian Restaurant" (2^{nd} level) than "Acehnese Restaurant" (4^{th} level) which is a successor of "Indonesian Restaurant". The idea behind the formulation of $DL(P)$ is to measure the length of the encodings of paths from the root to attributes $a \in P$ such that (1) these paths cover as few as possible attributes that are not in P, and (2) the paths are as short as possible. To construct the set of paths, we start from the root node a_1 and recursively move to the node children until reaching each attribute of P. Let S be the set of nodes of the paths. The cost associated to an attribute $a_i \in S$ is computed as follows:

1. If a_i is an internal node of the hierarchy and the end-point of a path, its encoding cost is $\log(\alpha_1)$, with α_1 is the probability that the path "*stops*" on that node. We encode with the value $\log(1-\alpha_1)$ the fact that a_i is an internal node of a path.

2. Let α_2 be the probability that $a_i \in P$ and $1 - \alpha_2$ the probability that a_i is not in P. In the case where $a_i \in P$ we must also encode the value $\hat{y}_i$. This requires $\log(D_x)$ bits, where $D_x \geq \hat{y}_1$ is an upper bound on values $\hat{y}_i$.

By multiplying these quantities with the respective cardinalities ($| \Uparrow P \setminus P|$ for the number of internal nodes of paths and $| \downarrow (\uparrow P)|$ for the end-points of the paths), the total encoding cost of P is:

$$\begin{aligned} DL(P) = &- | \Uparrow P \setminus P| \cdot \log(1 - \alpha_1) - | \downarrow (\uparrow P) \setminus L(\mathcal{H})| \cdot \log(\alpha_1) \\ &- | \downarrow (\uparrow P) \setminus P| \cdot \log(1 - \alpha_2) - |P| \cdot (\log(\alpha_2) + \log(D_x)) \,. \end{aligned}$$

α_1 and α_2 are user specified parameters. If the user prefers to have patterns P with deeper nodes from the hierarchy (nodes which are closer to the leaves), she can decrease the value of α_1. On the other hand, if she likes to have patterns P with larger number of attributes, she can increase α_2.

In what follows, we detail, as an example, the computation of $DL(\{a_4, a_5, a_6\})$ given in Fig. 6.2:

- Starting from the root a_1, we explore its children $\{a_2, a_6, a_9\}$. Then, we consider the node a_2 and explore its children. This requires a cost of $-2 \times \log(1 - \alpha_1)$ since there are two internal nodes on these paths.
- At that time, the description covers the attributes $\{a_3, a_4, a_5, a_6, a_9\}$ and thus contains P. Since a_6 and a_9 are not leaves, we need to specify that we stop on them. This takes a cost of $-2 \times \log(\alpha_1)$.
- We also specify that a_3, a_9 do not belong to P (with a cost of $-2 \times \log(1 - \alpha_2)$) and that a_4, a_5, a_6 belong to P (with a cost of $-3 \times \log(\alpha_2)$). Finally, we add the cost to communicate the $\hat{y}_i$ values for each attribute of P (with a cost of $3 \times \log(D_x)$).

The overal description length of P is then: $DL(P) = -2 \times \log(1 - \alpha_1) - 2 \times \log(\alpha_1) - 2 \times \log(1 - \alpha_2) - 3 \times (\log(\alpha_2) + \log(D_x))$.

6.3.3 Updating the background knowledge

When a rational user observes some patterns, her background knowledge may change to take into account these new pieces of information. It results that the observed patterns become expected by her and the background knowledge has to be updated as well. Let us denote by $\mathcal{O} \subseteq A$ the set of already observed nodes. The quality of a pattern P can thus be assessed using $SI(P|\mathcal{O})$, the subjective interestingness of P conditioned to the already observed values from $\mathcal{O}$:

$$SI(P|\mathcal{O}) = \frac{IC(P|\mathcal{O})}{DL(P)} = \frac{-\log(\Pr(P|\mathcal{O}))}{DL(P)}.$$

$\Pr(P|\mathcal{O})$ is the probability that P is present in the data given that the scales of attributes $a_i \in \mathcal{O}$ are equal to their observed values $\hat{y}_i$:

$$\Pr(P|\mathcal{O}) = \prod_{a_i \in P} \Pr(\hat{y}_i | \bigwedge_{a_j \in \mathcal{O}} \hat{y}_j).$$

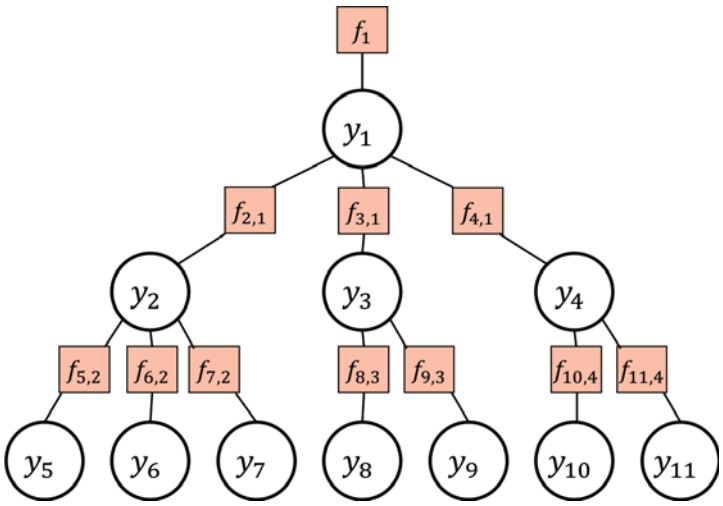


Figure 6.3: Factor graph of the tree in Fig. 6.1.

The conditional probabilities $\Pr(\hat{y_i}|\bigwedge_{a_j \in \mathcal{O}} \hat{y_j})$ can be computed using the conditional constraints between each x_i and its direct ancestor x_{π_i}, as expressed by Equation 6.2. These dependencies between variables Y_i constitute a *Bayesian tree* (i.e., a graphical model representation [32]) which states that the joint probability between two values of the tree is independent conditionally to the value of their direct ancestor. The probability of a given state $(y_1, ..., y_n)$ is thus:

$$\Pr(y_1, ..., y_n) = \prod_{i \in [\![1,n]\!]} \Pr(y_i|y_{\pi_i}).$$

The *sum-product inference* algorithm [32] can be used to update the background model. In order to use this algorithm, we first need to derive the corresponding *factor graph*. This can be constructed by making a factor function f_1 defined on Y_1, and a factor function f_{i,π_i} for each pair (y_i,y_{π_i}). The values of these functions are: $f_1(y_1) = \Pr(y_1)$, and $f_{i,\pi_i}(y_i, y_{\pi_i}) = \Pr(y_i|y_{\pi_i})$ for each (y_i, y_{π_i}). Then, the probability of a given state $(y_1, ..., y_n)$ of this graph is:

$$\Pr(y_1, ..., y_n) = f_1(y_1) \cdot \prod_{i \in [\![2,n]\!]} f_{i,\pi_i}(y_i, y_{\pi_i}).$$

Fig. 6.3 displays the factor graph of the hierarchy from Fig. 6.1.

The sum-product algorithm works only when the random variables have a finite number of states, while in our case y_i take their values in $\mathbb{N}$. To make the use of sum-product possible, we can limit the values of x_i to an upper bound $D_x > \max(\hat{x}_1, \bar{x}_1)$. If we set $D_x = 2^{20}$, x_i can variate from 0 to 2^{20}, and y_i from 0 to 20. For example, if values of $\bar{x}_i$ and $\hat{x}_i$ are all lower than 2^{15}, it can be sufficient to set $D_x = 2^{20}$, since $Pr(y_i > 20)$ will be extremely small and negligible. Before running sum-product, we prepare the values of f_1 for each $y_1 \in [\![0, \log(D_x)]\!]$, and the value of each f_{i,π_i} for each $(y_i, y_{\pi_i}) \in [\![0, \log(D_x)]\!]^2$:

- For the factor f_1, the computation can be done in constant time using the CDF of the geometric distribution:

$$f_1(y_1) = \Pr(2^{y_1} \leq x_1 < 2^{y_1+1}) = (1-p_1)^{2^{y_1}} - (1-p_1)^{2^{y_1+1}}.$$

- For the other factor functions, we have:

$$\begin{aligned} f_{i,\pi_i}(y_i, y_{\pi_i}) =& \Pr(y_i|y_{\pi_i}) = \frac{\Pr(y_i \wedge y_{\pi_i})}{\Pr(y_{\pi_i})}, \\ =& \frac{\sum_{x_{\pi_i}=2^{y_{\pi_i}}}^{2^{y_{\pi_i}+1}-1} \Pr(x_{\pi_i}) \cdot \Pr(2^{y_i} \leq x_i < 2^{y_i+1}|x_{\pi_i})}{\Pr(2^{y_{\pi_i}} \leq x_{\pi_i} < 2^{y_{\pi_i}+1})}. \end{aligned}$$

  And:

$$\begin{aligned} \Pr(2^{y_{\pi_i}} \leq x_{\pi_i} < 2^{y_{\pi_i}+1}) &= (1-p_i)^{2^{y_i}} - (1-p_i)^{2^{y_i+1}}, \\ \Pr(2^{y_i} \leq x_i < 2^{y_i+1}|x_{\pi_i}) &= I_{1-b_i}(x_{\pi_i} - 2^{y_i+1} - 1, 2^{y_i+1}) \\ &\quad - I_{1-b_i}(x_{\pi_i} - 2^{y_i} - 1, 2^{y_i}), \end{aligned}$$

  with $I_\alpha(m-k, k+1)$ is the regularized incomplete beta function which can be efficiently approximated [193].

The complexity for computing all the factor function values is in $\mathcal{O}(|A| \cdot D_x \cdot \log(D_x))$, and the memory complexity is $\mathcal{O}(|A| \cdot \log(D_x)^2)$. The computation of factor functions can be done once for the studied hierarchy, in order to be used repeatedly by sum-product each time we need to update the model. Sum-product can be applied as explained in [32]. The time complexity of this algorithm is $\mathcal{O}(|A| \cdot \log(D_x)^2)$, we recall that $|A|$ is the number of random variables Y_i, and $\log(D_x)^2$ is the number of possible values of each pair (y_i, y_{π_i}).

6.4 Finding the most interesting contrastive antichains

Finding contrastive antichains is computationally hard: the number of antichains in a rooted tree of order n is at most $2^{n-1}+1$, as shown by Klazar in [124]. Also, the interestingness of a contrastive antichain is a ratio of two measures, IC and DL, that vary unpredictably from one pattern to another and for which it is not straightforward to derive non-trivial bounds on their values to prune some uninteresting antichains. Nonetheless, we derive MICA-Miner (Algorithm 9), a heuristic algorithm based on a greedy strategy, which has a polynomial worst case complexity (i.e., $\mathcal{O}\left(|A|^2 \cdot \max(log(D_x)^2, |L(\mathcal{H})|)\right)$, see Section 6.4.1) and whose effectiveness is demonstrated experimentally. Before running this algorithm, we consider that the factor functions f_1 and f_{i,π_i} (for each $i \in [\![2, n]\!]$) are already computed based on the approach explained in Section 6.3.3.

MICA-Miner is an iterative algorithm that, at each iteration, updates the model using the sum-product algorithm for integrating the previously observed nodes $\mathcal{O}$. It then produces an antichain P based on

Algorithm 9: MICA-Miner

Input: $\mathcal{H}$: the hierarchical dataset, pre-computed values of factor functions f_1 and f_{i,π_i} for each $i \in [\![2, n]\!]$.

Output: $\mathcal{R}$: the antichains sorted based on iteratively updated SI.

```
R ← ⟨⟩
O ← ∅
repeat
    // Update the model with the sum-product algorithm
    // and derive the probabilities Pr(ŷ_i|O) for each y_i:
    Sum-product(H, O)
    // Compute an antichain with a greedy strategy:
    P ← GreedySearch(H)
    if P ≠ ∅ then
        R.append(P)
        O ← O ∪ P
until P = ∅;
```

GreedySearch. MICA-Miner continues until GreedySearch returns an empty pattern, which means that $IC(a) = 0 \; \forall a \in A$.

In order to greedily build an antichain, GreedySearch (Algorithm 10) starts from an empty pattern $P = \emptyset$ and a set of candidates $C = A$. It also uses a set Q that contains the current paths endpoints, i.e., Q contains the already known part of $\downarrow (\uparrow P)$, and Q will be equal to $\downarrow (\uparrow P)$ at the end of GreedySearch. In each step, GreedySearch chooses an attribute that will be added to Q and possibly to P, and removed from C (as well as its predecessors and successors). In order to decide which attribute to pick in each step, GreedySearch uses a heuristic $\frac{IC}{\overline{DL}}$ where $\overline{DL}$ is defined for an antichain P and a set Q s.t. $P \subseteq Q \subseteq \downarrow (\uparrow P)$. $\overline{DL}$ is similar to DL, the difference is that $\overline{DL}$ does not account for the cost of all the paths endpoints $\downarrow (\uparrow P)$, but only for Q, the already known part of the final $\downarrow (\uparrow P)$:

$$\begin{aligned}\overline{DL}(P, Q) = &- |\Uparrow P \setminus P| \cdot \log(1 - \alpha_1) - |Q \setminus L(\mathcal{H})| \cdot \log(\alpha_1) \\ &- |Q \setminus P| \cdot \log(1 - \alpha_2) - |P| \cdot (\log(\alpha_2) + \log(D_x)) \,.\end{aligned}$$

At each iteration, GreedySearch selects two attributes: $a^\star \in C$ and $f^\star \in \min(C)$ such that regarding to $\frac{IC}{\overline{DL}}$: (1) $a^\star$ is the best attribute to add to P , (2) $f^\star$ is the best attribute to add to $Q \setminus P$ (i.e., $f^\star$ is the best candidate to remove from C without add it to P). From $a^\star$ and $f^\star$, GreedySearch chooses the one that has the highest $\frac{IC}{\overline{DL}}$. If $a^\star$ is chosen, P is extended by this attribute. Otherwise, $f^\star$ and all its successors are removed from the set of candidates. When C becomes empty, the antichain P is returned.

6.4.1 Theoretical complexity of MICA-Miner

We derive the worst case complexity of MICA-Miner, which depends on the complexity of sum-product and GreedySearch. As stated in Section 6.3.3, sum-product has a worst case complexity of $\mathcal{O}(|A| \cdot log(D_x)^2)$ where $|A|$ is the number of attributes, and the parameter $D_x > max(\hat{x}_1, \bar{x}_1)$ is the considered number of possible values of x_i. In GreedySearch, each iteration costs at most $\mathcal{O}(|A|)$, and the maximum

Algorithm 10: GreedySearch

Input: $\mathcal{H}$: the hierarchical dataset, values of conditional probabilities $\Pr(\hat{y}_i|\mathcal{O})$ for each y_i
Output: P: A greedily constructed antichain

$P \leftarrow \emptyset, C \leftarrow A, Q \leftarrow \emptyset$
while $C \neq \emptyset$ **do**
  $a^\star \leftarrow \text{argmax}_{a \in C} \frac{IC(P \cup \{a\}|\mathcal{O})}{\overline{DL}(P \cup \{a\}, Q \cup \{a\})}$
  $f^\star \leftarrow \text{argmax}_{a \in \min(C)} \frac{IC(P|\mathcal{O})}{\overline{DL}(P, Q \cup \{a\})}$
  if $\frac{IC(P \cup \{a^\star\}|\mathcal{O})}{\overline{DL}(P \cup \{a^\star\}, Q \cup \{a^\star\})} \geq \frac{IC(P|\mathcal{O})}{\overline{DL}(P, Q \cup \{f^\star\})}$ **then**
    $P \leftarrow P \cup \{a^\star\}$
    $Q \leftarrow Q \cup \{a^\star\}$
    $C \leftarrow \{a \in C \mid a \not\preceq a^\star \wedge a^\star \not\preceq a\}$
  else
    $Q \leftarrow Q \cup \{f^\star\}$
    $C \leftarrow \{a \in C \mid f^\star \not\preceq a\}$
return P

number of iterations is equal to $|L(\mathcal{H})|$ which is the size of the largest antichain. Thus, GreedySearch has a worst case complexity of $\mathcal{O}(|A| \cdot |L(\mathcal{H})|)$. Since the maximum number of iterations in MICA-Miner is $|A|$, we conclude that its worst case complexity is $\mathcal{O}\left(|A|^2 \cdot \max(log(D_x)^2, |L(\mathcal{H})|)\right)$.

6.5 Experiments

In this section, we report our experimental results. These experiments aim to evaluate the performance and the quality of results provided by MICA-Miner in real world datasets. The method was implemented in Java and the experiments were performed on a machine equipped with Intel(R) Xeon(R) CPU @ 4.00GHz, and 128GB main memory, running Debian GNU/Linux 9.6. The code and the data are available[2].

Datasets and aims. We conduct experiments in three real world datasets whose main characteristics are given in Table 6.1:

- **FV**: Foursquare venues [77], the nodes a_i in this hierarchical dataset are categories of venues[3] (food, restaurant, event, etc.). We aim to study the distribution of venue categories in Amsterdam and London, and discover which kind of categories are surprisingly frequent/rare in each of these cities. We then consider two case studies: (1) **FV-Ams** for Amsterdam: the observed values $\hat{x}_i$ are the values of categories in Amsterdam, (2) **FV-Lon** for London: the observed values $\hat{x}_i$ are the values of categories in London. For both cases, the priors $\bar{x}_i$ are the aggregation of categories values throughout 20 well known cities from Europe and USA (London, Barcelona, New York, etc.).
- **OFF**: Open Food Facts[4] is a free food product database that gathers information and data on food products around the world. Each product is described by its food category (Plant-based food, ...) and its country. The categories represent the nodes a_i of the hierarchy. We want to compare the distribution

[2] http://goo.gl/6Vx5dz
[3] https://developer.foursquare.com/docs/resources/categories
[4] https://world.openfoodfacts.org/

Datasets	$\|A\|$	$\|\hat{x}_1\|$	# levels	branching factor
FV-Ams	846	9030	5	13
FV-Lon	846	25220	5	13
OFF	2000	147,411	10	3.6
EP_Abs	358	$[7470; 30, 673]$	5	5.17

Table 6.1: Description of the real-world datasets.

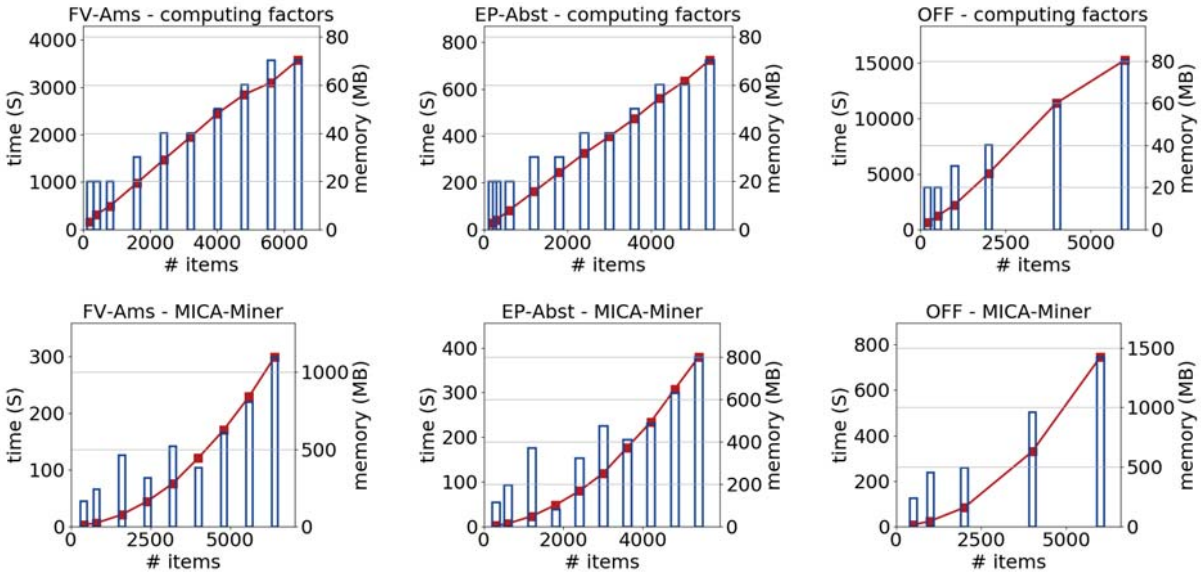


Figure 6.4: Time (line) and memory consumption (bars) of the factor functions computation (line 1), and MICA-Miner (line 2) in the studied datasets w.r.t. $|A|$.

of food categories between France and USA, the two countries with the most contributions. Then, the values $\hat{x}_i$ (resp. $\bar{x}_i$) are the number of products for each category in France (resp. USA).

- **EP_Abst**: European Parliament[5] is a dataset that depicts ballots of the European Parliament between 2014 and 2019. Each ballot is described by the corresponding topics and the votes (for, against, abstain) of each deputy, whose political groups are given. Topics are organized following a hierarchy (e.g., the topic "5.10 Economic Union" is the strict predecessor of "5.10.02 Price policy, price stabilisation"). We build a hierarchical dataset for each political group where the values $\hat{x}_i$ correspond to the number of abstentions: the number of times a deputy from this group abstains in the topic a_i. The priors $\bar{x}_i$ are constructed from the total number of ballots by topic.

In this experimental study, we aim to answer the following questions: How much time and memory does MICA-Miner consume in practice? How does the size of the hierarchy $|A|$ impact the performance of MICA-Miner? What is the impact of model updating on the found antichains? Are the antichains found by MICA-Miner relevant and informative?

Quantitative evaluation. We study the runtime and memory consumption for (1) the computation of

[5] `http://parltrack.euwiki.org/`

factor functions (see Section 6.3.3), and (2) MICA-Miner.

We vary the number of attributes in the real world datasets as follows: in order to decrease the number of attributes, we iteratively remove leaves that are the furthest from the root, and in order to increase the number of attributes, we duplicate the tree as many times as we need and then we add a new root that links all the duplications. Results are given in Fig. 6.4, line 1 for the factor functions, and line 2 for MICA-Miner. As it was theoretically expected, the computation of factors requires a time and memory that linearly increase with the number of attributes. Moreover, the time increases for MICA-Miner in a quadratic manner (a fitted model $a \cdot |A|^2 + b$ approximates the time with an average square error below $7(s^2)$), its memory consumption also follows the same quadratic trend.

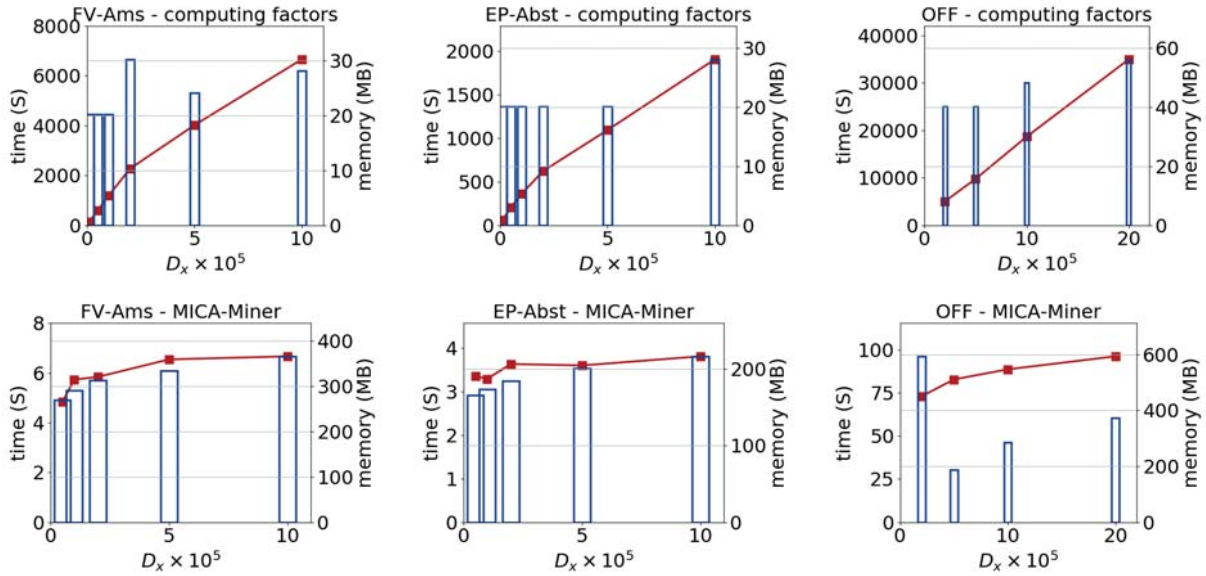


Figure 6.5: Time (line) and memory consumption (bars) of the factor functions computation (row 1), and MICA-Miner (row 2) in the studied datasets w.r.t. D_x.

We also study the influence of the parameter D_x on the runtime and memory consumption. D_x is an upper bound on the value of variables x_i, it needs to cover the used values $\bar{x}_i$ and $\hat{x}_i$, this means that $D_x \geq max(\hat{x}_i, \bar{x}_i)$. Fig. 6.5 reports the results. The runtime grows linearly w.r.t D_x for the pre-processing step while the increasing of D_x does not significantly impact the runtime of MICA-Miner. The memory consumption slightly increases in general w.r.t. D_x in both cases.

Comparative evaluation. There is no approach in the literature that supports the discovery of subjectively interesting antichains in hierarchies. Nevertheless, we identify some baselines that we consider in our study to highlight the characteristics of the patterns found with MICA-Miner: (1) SI *without update* which returns the best results according to the SI measure without updating the background knowledge model; (2)*WRAcc* that uses the $WRAcc$ measure [130]. This measure has been largely used in Subgroup Discovery in order to evaluate the prevalence of a given class of objects in a specific subset of data (a subgroup). We adapted it to our problem in order to evaluate the prevalence/absence of nodes $a \in P$

Figure 6.6: The first pattern based on the $WRAcc$ measure in the **FV-Ams** dataset.

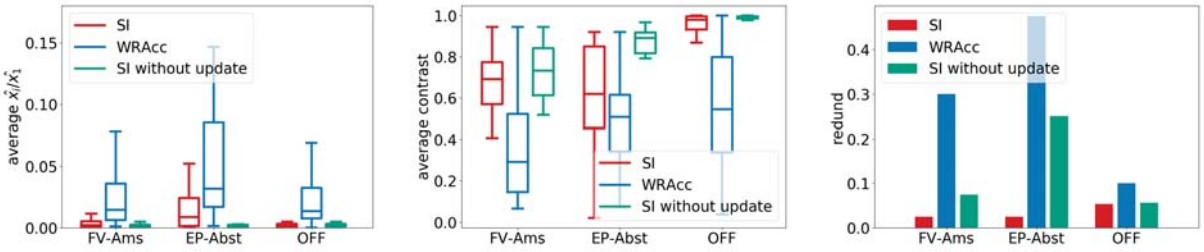


Figure 6.7: Comparison of (1) normalized supports, (2) contrast, (3) redundancy between the top antichains of the three measures: SI, $WRAcc$, and SI without model updating.

in the studied dataset. We consider that a subgroup is defined with the antichain P, the size of positive class is $\hat{x}_i$, and the size of negative class is $\bar{x}_i$.

We compare the following properties computed in the top k patterns returned in each configuration:

- **Average normalized values of attributes**: we show the average values of $\frac{\hat{x}_i}{\hat{x}_1}$ corresponding to nodes in the top patterns.

- **Average contrast**: we want to evaluate how much the values of $\hat{x}_i$ are contrastive comparing with their corresponding $\bar{x}_i$. We propose to use the following contrast measure defined for an antichain $P \subseteq E$: $contrast(P) = \sum_{a_i \in P} \frac{max(\hat{x}_i, \bar{x}_i) - min(\hat{x}_i, \bar{x}_i)}{max(\hat{x}_i, \bar{x}_i)}$.

- **Redundancy**: An important goal of the iterative updating of the model is to avoid communicating to the user similar information several times. For example, after informing the user that a_i ="Restaurant" is prevalent, it is less informative to tell her that a_{π_i} ="Food" is also prevalent. We aim to measure this kind of redundancy between patterns returned by each approach based on the following measure defined for a set of patterns R:

$$redund(R) = \frac{\sum_{P,P' \in R, P \neq P'} |\{a_i \in P | \exists a_{\pi_i} \in P' : (\hat{x}_i - \bar{x}_i) \cdot (\hat{x}_{\pi_i} - \bar{x}_{\pi_i}) \geq 0\}|}{|R| \times \sum_{P \in R} |P|}.$$

Fig. 6.7 reports the values of these properties in each of the three configurations (1) SI (our approach), (2) $WRAcc$ measure, and (3) SI without update. These properties are computed based on the top 5 patterns in **FV-Ams** and **EP-Abst**, and on top 20 in **OFF** (since it is a much larger hierarchy). The average values of attributes in $WRAcc$ results are significantly large, its top patterns generally contains

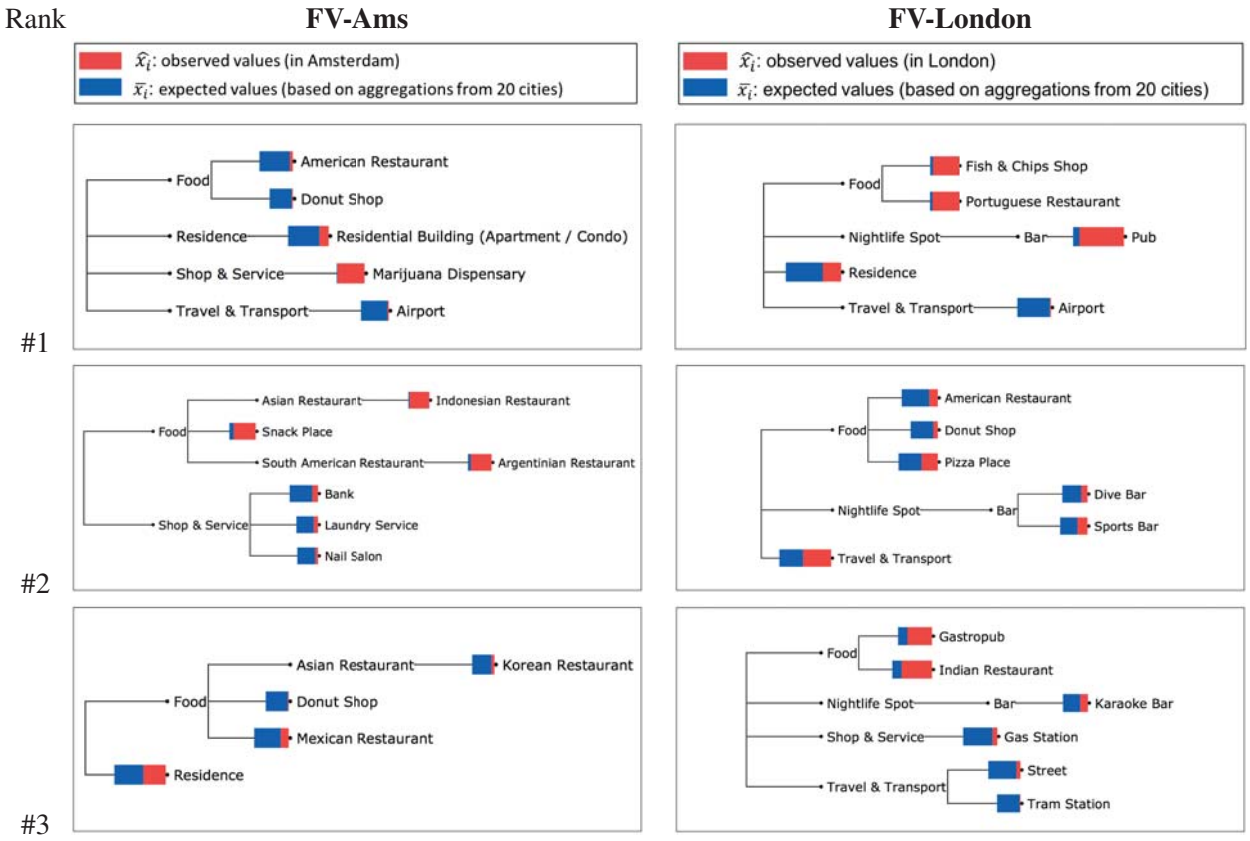


Figure 6.8: Top 3 antichains in Amsterdam and London (based on the prior beliefs over 20 cities).

attributes with the highest observed values in the dataset. However, its average contrast is remarkably low. Fig. 6.6 shows the first pattern found by $WRAcc$ in **FV-Ams**. Most of its nodes are not significantly contrastive, i.e., there is not a high difference between the observed value and the expected value. We can also notice that the values of $redund$ are the lowest for the results of SI, and those of the $WRAcc$ have the highest $redund$. The value of $redund$ for SI is clearly lower than its value for "SI without update" in **FV-Ams** and **EP-Abst**. The difference is not remarkable for the **OFF** dataset, indeed, since it is a larger dataset, there is less chance to have redundancies in the results. To sum up, our method allows to discover more contrastive antichains than methods that do not take into account the prior beliefs. Furthermore, updating the background knowledge at each iteration makes it possible to provide less redundant results.

Illustrative results. We report in Figures 6.8 and 6.9 the top 3 contrastive antichains discovered by MICA-Miner in **FV-Ams**, **FV-Lon**, and **OFF** datasets. The blue color quantifies the expected value by the user (based on her beliefs over 20 cities), and the red color is the observed attribute value in Amsterdam (resp. London). The first antichain in Amsterdam informs that American Restaurants, Donut Shops, Residential Buildings, and Airports are much less frequent than expected, while Marijuana Dispensary is significantly over-expressed. This last type of venue is indeed characteristic of Amsterdam, because it is authorized in this city while it is generally illegal in other cities. London is characterized

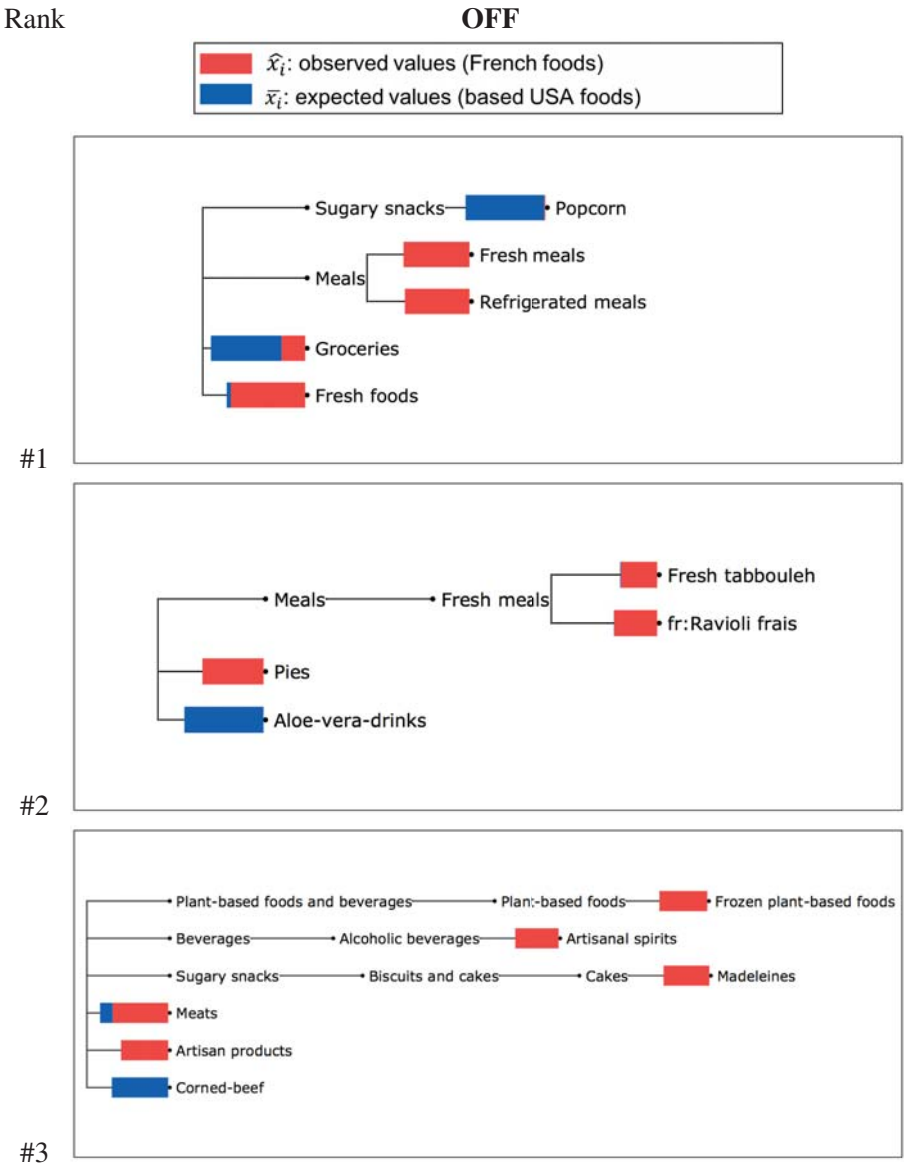


Figure 6.9: Top 3 antichains in Open Food Fact dataset (comparison of French products and USA products).

with a high number of Fish & Chip Shops, Portuguese Restaurants, and Pubs, and an under-expression of Residences and Airports. We point out that Airport venues can correspond to any place related to Airport (Airport Food Court, Airport Gate, Airport Tram, etc.). After assimilating this first antichain, the user will change her expectations about the rest of the data. For example, she may assume that in general there is a high presence of Food venues and Bars in London, which is also incorporated in our model using the conditional expectation. Due to this hypothesis, the second antichain will notify the user that despite of the high observed values in the first antichain, there is some specific types of Food venues and Bars that are under expressed (American Restaurant, Donut Shop, etc.). For **OFF** dataset,

the Red color quantifies the observed counts of products from France, and blue color corresponds to the expected values derived from products distributions from USA. The first antichain can be interpreted as: *From an American point of view, the French food is characterized with a high number of fresh meals, refrigerated meals, fresh foods, and a low presence of groceries and popcorn products*.

6.6 Conclusion

We have extended the subjective interestingness framework proposed in [59] to consider the case where attributes are hierarchically organized. In practice, beliefs of a user about attributes of a hierarchy influence each other. In other terms, informing the user about the value of an attribute a_i will bias her expectations about values of other attributes, especially the parent and children of a_i. In order to take into account this dependency, the model that we propose in this chapter exploits the hierarchical relation to both propagate the beliefs and to update the background knowledge. We have used this model to define the novel problem of mining contrastive antichains in hierarchically organized sets of attributes. We have proposed a greedy algorithm that efficiently and iteratively mine contrastive antichains. Extensive empirical results on several real-world datasets demonstrate the performance and the effectiveness of this approach.

In this work, we have studied the hierarchy structure of attributes independently from any graph structure. In fact, the problem of incorporating the dependencies induced by the hierarchy in the interestingness model is already complex on its own. A promising avenue of this work is to integrate this model into SIAS-Miner (see Chapter 4), to mine vertex-attributed graphs where a hierarchical relationship stands between attributes. Or more generally, this model can be used to assess the interestingness of patterns in the Subgroup Discovery task.

CHAPTER 7

Conclusion and Future Directions

7.1 Conclusion

We have tackled the problem of mining useful patterns in vertex-attributed graphs, i.e., graphs augmented with attributes describing vertices. We have proposed new pattern languages, interestingness measures, and associated mining algorithms for this problem. Specifically, the main questions that we have considered are: (1) how to identify patterns with discriminative features in vertex-attributed graphs? (2) how to assess the quality of the patterns ?, especially how to take into account the users in such assessment ? (3) when computing the interestigness, how to take benefit from domain knowledge such as hierarchies of attributes?

Chapter 3 aims to answer question (1). To this end, we have introduced the new problem of Exceptional Attributed Subgraph Mining. This task is concerned with identifying connected subgraphs having some attribute values that are exceptionally high or low. This problem is rooted on Subgroup Discovery [130] and Exceptional Model Mining [71] frameworks. Indeed, we assess the quality of patterns by contrasting their local subgraph model with the global model of the whole graph. Both local and global models are computed on a same subset of attributes that is enumerated by the mining algorithm. We have defined two algorithms to mine such patterns. First, we have designed an exhaustive method that extracts the complete set of closed exceptional subgraphs. Second, we have proposed a heuristic algorithm to sample the output space of closed patterns to enable time-budget analysis. Experiments have demonstrated the performance of the proposed approach and its ability to identify relevant exceptional subgraphs. However, this approach does not make it possible to answer questions (2) and (3). In fact, the user, her preferences, and her priors are not taken into account. Furthermore, the assimilation cost of the patterns by the user is not considered. Consequently, the proposed patterns may be uninformative (for the user) or too difficult to interpret. Chapters 4 and 5 build on these limitations by proposing methods that aims to take into account the user in the mining process.

Chapter 4 presents a vertex-attributed graph mining approach rooted on the Subjective Interestingness framework defined by De Bie [59]. This method incorporates user's background knowledge about the data, and accounts for the assimilation cost of patterns. The interestingness is defined based on information theory, as the ratio of the information content (IC) over the description length (DL). The IC is the amount of information provided by showing the user a pattern. The quantification is based on the gain from a Maximum Entropy background model that delineates the current knowledge of a user. The DL assesses the complexity of reading a pattern, the user being interested in concise and intuitive descriptions. Thus, we have proposed to redescribe a set of vertices as an intersection of neighborhoods of certain distance from certain vertices, the distance and vertices making up the description of the subgraph. Based on this model, we have designed an algorithm that extracts the set of exceptional attributed subgraphs that are both informative and easy to assimilate. We have also shown how this interestingness model can be updated each time a pattern is presented to the user. This updating technique allows to continuously capture patterns that are not redundant, and that provide new information comparing with the already acquired one from previous patterns.

In Chapter 5, we have proposed another user-driven interestingness for vertex-attributed graphs. This interestingness aims at integrating user's preferences when assessing the quality of patterns. These preferences are infered from an interactive process where the user is asked to select patterns that she likes. Based on this measure, we have defined an exceptional attributed subgraph mining method for the task of event detection in social media such as Twitter. This choice is mainly motivated by the assessment of the method. In fact, a large number of people can understand and interpret results of event detection in social media, which makes it more realistic to test and evaluate this approach. In the empirical evaluation, we have first compared this approach with state of the art event detection methods in a data-driven setting, i.e., without integrating user feedbacks. These experiments have demonstrated that the runtime of our approach outperforms state of the art methods by a factor of two to several orders of magnitude, and that the proposed approach is more robust to noise. Second, we have demonstrated the ability of this method to integrate user's preferences, through experiments with real users hired from a Crowdsourcing platform.

Attributes that describe vertices are often organized as a hierarchy. For example, throughout experiments of Chapters 4 and 5, we have employed a graph that depicts areas of a city with the prevalence of different kinds of facilities (food places such as restaurants, colleges, etc.). A hierarchy relation is available about types of facilities, e.g., the attribute "restaurant" is a subtype of the "food", and the attribute "French restaurant" is a subtype of "restaurant". This hierarchy imposes particular constraints on the values of related attributes—e.g. there cannot be more French restaurants than food venues. Moreover, knowing that a city has many food venues makes it less surprising that it also has many French restaurants, and vice versa. In Chapter 6, we have proposed an approach that explicitly integrates this domain knowledge in the subjective interestingness [59] used to evaluate patterns defined on hierarchical attributes. This model accounts for the inferences that the user would make when attributes are hierarchical, which allows to avoid redundancy of information related to these dependencies. We have

shown in Chapter 6 how it can be exploited to mine the so-called contrastive antichains, i.e., particular kinds of subsets of their attributes and their values. We have addressed the problem of assessing the interestingness of a contrastive antichain, in the sense that it concisely describes the unique aspects of a dataset, and this while duly taking into account the known attribute dependencies implied by the hierarchy. This approach is capable of accounting for previously identified contrastive antichains, making iterative mining possible.

7.2 Future directions

We have addressed different questions related to the problem of mining vertex-attributed graphs. Several interesting results have been achieved, but many problems remain, and new questions come to mind after exploring this subject. In what follows, we will discuss these problems and the different future directions that they can imply.

Alternative feedback formats. The interactive approach proposed in Chapter 5 integrates the user feedback by asking her to select patterns that she likes. The feedback formats can be extended to let the user express her point of view about the results in more diverse ways. For example, we can include the possibility of disliking patterns, or assigning a numerical score (rating them) [31]. However, when more feedback choices are offered to the user, the interactive process becomes more complex. A challenge is to find the good balance between the capacity of the feedback to represent user's preferences and the complexity of interaction.

Alternative priors types. The Subjective Interestingness framework of De Bie [59] requires to express the background knowledge as constraints for the Maximum Entropy optimization problem. In Chapter 4, we have considered the attributes and vertices margins (sums) as priors. In Chapter 6, we have shown how to incorporate the attribute hierarchy in the model. For both of these works, we have also presented how to update the model when a given pattern is already a part of the user's knowledge. However, several other types of priors exist. For instance, attributes can be organized as directed acyclic graphs instead of simple hierarchies. Another example of priors is when the user expects that two attributes are highly correlated. How to include such priors into the model? After coming up with solutions for the types of priors that occur the most in real life, an interesting avenue is to design a tool that: (1) allows the user to easily express her priors, (2) encode them as constraints into the model, and (3) derive the corresponding model efficiently.

Integrating both prior knowledge and preferences. We have studied separately the incorporation of user's background knowledge and preferences in the vertex-attributed graph mining problem. In fact, each of this subjects is challenging on it own and requires significant effort of investigation. A natural question that emerges is: how to consider both of them into the same model? such a model would allow to identify patterns that provide to the user new information that falls within her topics of interest.

Consequently, this would improve the quality of vertex-attributed graph mining results. This avenue is challenging as each of the two purposes can alter the other one. In fact, integrating user's background knowledge leads to discovering something new and different, while integrating her preferences produces patterns containing features that are similar to those of the already found ones. How to discover patterns that contain features of interest for the user, but that are new and unexpected?

Presenting patterns to the user. When it comes to complex pattern languages such as attributed subgraphs, an important question needs to be addressed: how to communicate the pattern to the user? Often, the size of the set of vertices $U \subseteq V$ associated to a subgraph pattern is huge. When this pattern is presented, specifying each vertex of U to the user is overwhelming. In Chapter 4, we have proposed a way to redescribe a set U by the intersection of neighborhoods of a much smaller set of vertices. For example in Twitter, this techniques is able to redescribe a community U as "the common followers of two famous persons X and Y". Several alternative redescriptions can still be investigated according to the applications, such as spanning trees (network) and paths (biological networks). The use of these redescriptions can be combined with sophisticated visualization techniques in order to provide a pattern rendering that eases the task of data analysis.

Algorithmic aspects. We have proposed algorithms to mine different kinds of patterns defined in this thesis. Some of these algorithms are exhaustive and extract the exact set of results. Although these methods are optimized using closed structures enumeration techniques and efficient upper bounds, their scalability remains limited as the search space size is exponential in the size of the dataset. We have also proposed some heuristic algorithms based on strategies such as output space sampling and gready search. While these methods scale better in practice, they don't provide any theorethical guarantee about the quality of results. An interesting future direction is to investigate new vertex-graph mining algorithms that are anytime, or that are heuristic with theorethical guarantees. Particularly, anytime algorithms have proven their efficiency in several data mining problems such as Time Series Motif Discovery [232] and Subgroup Discovery [22, 43].

More generic attributed graph structures In this thesis, we have focused on graphs where vertex-attributes are counts (non-negative integers). More complex models include attributes with diverse types (Boolean, categorical, real numbers, etc.). Proposing pattern mining approaches to mine these structures makes it possible to handle a much larger number of real world datasets. For instance, De Bie [60] shows how to incorporate each of Boolean, real and natural attributes in the subjective interestingness model for mining tiles in binary databases. These results can be extended to propose generic models for mining vertex-attributed graphs.

Mining uncertain (probabilistic) graphs. A special type of graphs is when uncertainty is introduced in the definition of their vertices, edges, or attributes. For example, an uncertain plain graphs $G = (V, E, p)$ can be such that $\forall e \in E$, $p(e)$ is the probability that e exists. These graphs occur in many

application domains where data management systems are required to deal with uncertainty in interrelated data, such as computational biology, social network analysis, network reliability. Also, mining this kind of graphs has been highly motivated by the challenge of privacy preserving, particularly after the adoption of the General Data Protection Regulation (GDPR) law by the European Parliament in May 2018. For instance, instead of sharing a graph that depics the exact friendship relations in a social network, the existence of friendships can be communicated as probabilities. While this graph makes it possible to preserve individual information, it still enables mining patterns defined on communities, e.g., mining the densest subgraph [233] and frequent subgraphs [234, 235], clustering [52, 103], etc. Mining this kind of graphs is a challenging topic. In fact, many problems that are relatively easy to solve in exact graphs become very difficult (or even intractable) in uncertain graphs. Although the mining of probabilistic graphs has received a great attention when it comes to plain graphs without attributes [52, 103, 233–235], mining uncertain attributed graphs (e.g., vertex-attributed graphs) has not been sufficiently studied. This category of graphs enables a more realistic representation of attributed graphs when data are uncertain. Interestingly, some recent works have been proposed to mine global models (e.g., clustering) in vertex-attributed graphs [139]. A promissing future direction is to address the problem of mining local patterns in uncertain attributed graphs.

Pattern mining to interpret black box models. The last decade has witnessed the rise of ubiquitous opaque decision systems (black box models), e.g., Deep Neural Networks (DNNs). While many of these systems are able to achieve high prediction accuracy, their internal decision process is often inexplicable. In other terms, when these models predict a specific output, we are not able to understand what makes them give such a result. This lack of explanation constitutes both a practical and an ethical issue. Indeed, understanding automatic prediction results is crutial in many applications , e.g., safety-critical industries such as self-driving cars, robotic assistants, and personalized medicine. For this reason, many scientific communities strive to provide interpretable machine learning decision models [97], sometimes at the cost of scarifying accuracy. An interesting avenue in this matter is to investigate the usability of pattern mining techniques to explain black box models, i.e., finding patterns that occur in these models in different contexts to understand their behavior. For example, when it comes to models that can be represented as graphs (e.g., DNNs), interpretable machine learning would take advantage from the advances achieved in the topic of pattern mining in graphs and attributed graphs.

Bibliography

[1] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. Eventweet: Online localized event detection from twitter. *Proc. VLDB Endow.*, 6(12):1326–1329, August 2013. ISSN 2150-8097. (cited on page: 89).

[2] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. Eventweet: Online localized event detection from twitter. *PVLDB*, 6(12):1326–1329, 2013. (cited on page: 88).

[3] James Abello, Mauricio G. C. Resende, and Sandra Sudarsky. Massive quasi-clique detection. In *LATIN 2002: Theoretical Informatics, 5th Latin American Symposium, Cancun, Mexico, April 3-6, 2002, Proceedings*, pages 598–612, 2002. (cited on page: 13).

[4] Florian Adriaens, Jefrey Lijffijt, and Tijl De Bie. Subjectively interesting connecting trees. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part II*, pages 53–69, 2017. (cited on page: 25).

[5] C.-C. Aggarwal and K. Subbian. Event detection in social streams. In *SIAM DM*, pages 624–635, 2012. (cited on page: 88).

[6] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 3–14, 1995. (cited on page: 2).

[7] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993.*, pages 207–216, 1993. (cited on pages: 1, 3).

[8] Rezwan Ahmed and George Karypis. Algorithms for mining the evolution of conserved relational states in dynamic networks. *Knowl. Inf. Syst.*, 33(3):603–630, 2012. (cited on page: 14).

[9] Luca Maria Aiello, Georgios Petkos, Carlos Martin, David Corney, Symeon Papadopoulos, Ryan Skraba, Ayse Goker, Ioannis Kompatsiaris, and Alejandro Jaimes. Sensing trending topics in twitter. *Trans. Multi.*, 15(6):1268–1282, 2013. (cited on page: 88).

[10] Mohammad Akbari, Xia Hu, Liqiang Nie, and Tat-Seng Chua. From tweets to wellness: Wellness event detection from twitter streams. In *AAAI*, pages 87–93. AAAI Press, 2016. (cited on page: 87).

[11] E. A. Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM J. Comput.*, 2(1): 1–6, 1973. (cited on pages: 2, 10).

[12] Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. PICS: parameter-free identification of cohesive subgroups in large attributed graphs. In *Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012.*, pages 439–450, 2012. (cited on pages: 10, 16).

[13] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *CoRR*, cond-mat/0106096, 2001. (cited on page: 11).

[14] John M. Aronis, Foster J. Provost, and Bruce G. Buchanan. Exploiting background knowledge in automated discovery. In *KDD*, pages 355–358, 1996. (cited on page: 116).

[15] Martin Atzmueller, Stephan Doerfel, and Folke Mitzlaff. Description-oriented community detection using exhaustive subgroup discovery. *Inf. Sci.*, 329:965–984, 2016. (cited on pages: 11, 23, and 29).

[16] Martin Atzmüller and Frank Puppe. Sd-map - A fast algorithm for exhaustive subgroup discovery. In *Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18-22, 2006, Proceedings*, pages 6–17, 2006. (cited on page: 22).

[17] Gary D. Bader and Christopher W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4:2, 2003. (cited on page: 10).

[18] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific american*, 288(5): 60–69, 2003. (cited on page: 11).

[19] Stephen D. Bay and Michael J. Pazzani. Detecting group differences: Mining contrast sets. *Data Min. Knowl. Discov.*, 5(3):213–246, 2001. (cited on page: 21).

[20] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. In *ICWSM*, 2011. (cited on page: 87).

[21] Tim De Beéck, Arjen Hommersom, Jan Van Haaren, Maarten van der Heijden, Jesse Davis, Peter Lucas, Lucy Overbeek, and Iris Nagtegaal. Mining hierarchical pathology data using inductive logic programming. In *AIME*, pages 76–85, 2015. (cited on page: 116).

[22] Aimene Belfodil, Adnene Belfodil, and Mehdi Kaytoue. Anytime subgroup discovery in numerical domains with guarantees. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part II*, pages 500–516, 2018. (cited on pages: 4, 22, and 140).

[23] Ahmed Anes Bendimerad, Marc Plantevit, and Céline Robardet. Unsupervised exceptional attributed sub-graph mining in urban data. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 21–30, 2016. (cited on pages: 7, 44, 48, and 90).

[24] Ahmed Anes Bendimerad, Marc Plantevit, and Céline Robardet. Mining exceptional closed patterns in attributed graphs. *Knowl. Inf. Syst.*, 56(1):1–25, 2018. (cited on pages: 7, 74, and 76).

[25] Anes Bendimerad, Ahmad Mel, Jefrey Lijffijt, Marc Plantevit, Céline Robardet, and Tijl De Bie. Mining subjectively interesting attributed subgraphs. In *Proceedings of the 14th Workshop on Mining and Learning with Graphs, MLG '18, London, United Kingdom, Aug 20, 2018*, 2018. (cited on page: 8).

[26] Anes Bendimerad, Jefrey Lijffijt, Marc Plantevit, Céline Robardet, and Tijl De Bie. Contrastive antichains in hierarchies. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, Alaska - USA, August 4 - 8, 2019*, 2019. (cited on page: 8).

[27] Anes Bendimerad, Marc Plantevit, Celine Robardet, and Sihem Amer-Yahia. User-driven geolocated event detection in social media. *IEEE Transactions on Knowledge and Data Engineering*, 2019. (cited on page: 8).

[28] Michele Berlingerio, Francesco Bonchi, Björn Bringmann, and Aristides Gionis. Mining graph evolution rules. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part I*, pages 115–130, 2009. (cited on page: 14).

[29] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings.*, pages 217–235, 1999. (cited on page: 17).

[30] M. Bhuiyan and M. Al Hasan. Interactive knowledge discovery from hidden data through sampling of frequent patterns. *Statistical Analysis and Data Mining*, 9(4):205–229, 2016. (cited on pages: 3, 27, and 97).

[31] Mansurul Alam Bhuiyan and Mohammad Al Hasan. PRIIME: A generic framework for interactive personalized interesting pattern discovery. In *2016 IEEE International Conference on Big Data,*

BigData 2016, Washington DC, USA, December 5-8, 2016, pages 606–615, 2016. (cited on page: 139).

[32] Christopher M. Bishop. *Graphical models*. Pattern recognition and machine learning. 2007. (cited on pages: 126, 127).

[33] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008. (cited on pages: 91, 99).

[34] Brigitte Boden, Stephan Günnemann, Holger Hoffmann, and Thomas Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 1258–1266, 2012. (cited on page: 14).

[35] Aleksandar Bojchevski and Stephan Günnemann. Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2738–2745, 2018. (cited on pages: 16, 19, and 20).

[36] Mario Boley, Tamás Horváth, Axel Poigné, and Stefan Wrobel. Listing closed sets of strongly accessible set systems with applications to data mining. *Theor. Comput. Sci.*, 411(3):691–700, 2010. (cited on page: 69).

[37] Mario Boley, Claudio Lucchese, Daniel Paurat, and Thomas Gärtner. Direct local pattern sampling by efficient two-step random procedures. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 582–590, 2011. (cited on pages: 4, 22, 32, and 40).

[38] Mario Boley, Sandy Moens, and Thomas Gärtner. Linear space direct pattern sampling using coupling from the past. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 69–77, 2012. (cited on pages: 4, 22).

[39] Francesco Bonchi and Claudio Lucchese. Extending the state-of-the-art of constraint-based pattern discovery. *Data Knowl. Eng.*, 60(2):377–399, 2007. (cited on page: 3).

[40] Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Antti Ukkonen. Chromatic correlation clustering. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 1321–1329, 2012. (cited on page: 14).

[41] Karsten M. Borgwardt, Hans-Peter Kriegel, and Peter Wackersreuther. Pattern mining in frequent dynamic subgraphs. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*, pages 818–822, 2006. (cited on page: 14).

[42] Guillaume Bosc. *Anytime discovery of a diverse set of patterns with Monte Carlo tree search. (Découverte d'un ensemble diversifié de motifs avec la recherche arborescente de Monte Carlo)*. PhD thesis, University of Lyon, France, 2017. (cited on pages: 22, 23).

[43] Guillaume Bosc, Jean-François Boulicaut, Chedy Raïssi, and Mehdi Kaytoue. Anytime discovery of a diverse set of patterns with monte carlo tree search. *Data Min. Knowl. Discov.*, 32(3):604–650, 2018. (cited on pages: 4, 22, and 140).

[44] Cécile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenková. Clustering attributed graphs: Models, measures and methods. *Network Science*, 3(3):408–444, 2015. (cited on pages: 10, 13, and 16).

[45] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Discov.*, 7(1):5–22, 2003. (cited on page: 3).

[46] Jean-François Boulicaut, Marc Plantevit, and Céline Robardet. Local pattern detection in attributed graphs. In *Solving Large Scale Learning Tasks. Challenges and Algorithms - Essays Dedicated to Katharina Morik on the Occasion of Her 60th Birthday*, pages 168–183, 2016. (cited on page: 44).

[47] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. (cited on page: 26).

[48] Björn Bringmann and Siegfried Nijssen. What is frequent in a single graph? In *Advances in Knowledge Discovery and Data Mining, 12th Pacific-Asia Conference, PAKDD 2008, Osaka, Japan, May 20-23, 2008 Proceedings*, pages 858–863, 2008. (cited on page: 2).

[49] Coenraad Bron and Joep Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973. (cited on pages: 2, 10, and 12).

[50] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, et al. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic acids research*, 31(9):2443–2450, 2003. (cited on page: 13).

[51] Huiping Cao, Nikos Mamoulis, and David W. Cheung. Mining frequent spatio-temporal sequential patterns. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 82–89, 2005. (cited on page: 2).

[52] Matteo Ceccarello, Carlo Fantozzi, Andrea Pietracaprina, Geppino Pucci, and Fabio Vandin. Clustering uncertain graphs. *PVLDB*, 11(4):472–484, 2017. (cited on page: 141).

[53] Feng Chen and Daniel B. Neill. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. In *KDD '14*, pages 1166–1175, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2956-9. (cited on page: 88).

[54] Ling Chen and Abhishek Roy. Event detection from flickr data through wavelet-based spatial analysis. In *CIKM*, pages 523–532, 2009. (cited on page: 88).

[55] Yun Chi, Yirong Yang, and Richard R. Muntz. Indexing and mining free trees. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, pages 509–512, 2003. (cited on page: 2).

[56] Yun Chi, Richard R. Muntz, Siegfried Nijssen, and Joost N. Kok. Frequent subtree mining - an overview. *Fundam. Inform.*, 66(1-2):161–198, 2005. (cited on page: 2).

[57] Moonjung Cho, Jian Pei, Haixun Wang, and Wei Wang. Preference-based frequent pattern mining. *IJDWM*, 1(4):56–77, 2005. (cited on page: 27).

[58] Thomas M Cover and Joy A Thomas. Entropy, relative entropy and mutual information. *Elements of information theory*, 2:1–55, 1991. (cited on pages: 56, 61, and 124).

[59] Tijl De Bie. An information theoretic framework for data mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 564–572, 2011. (cited on pages: ix, 3, 6, 7, 11, 24, 25, 29, 54, 56, 60, 61, 135, 138, and 139).

[60] Tijl De Bie. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Min. Knowl. Discov.*, 23(3):407–446, 2011. (cited on pages: 3, 25, 26, 61, 63, 121, 122, 123, 124, and 140).

[61] Cláudio Rebelo de Sá, Wouter Duivesteijn, Paulo J. Azevedo, Alípio Mário Jorge, Carlos Soares, and Arno J. Knobbe. Discovering a taste for the unusual: exceptional models for preference mining. *Machine Learning*, 107(11):1775–1807, 2018. (cited on page: 22).

[62] María José del Jesús, Pedro González, Francisco Herrera, and Mikel Mesonero. Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. *IEEE Trans. Fuzzy Systems*, 15(4):578–592, 2007. (cited on pages: 4, 22).

[63] Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. (cited on pages: 108, 110, and 111).

[64] Elise Desmier, Marc Plantevit, Céline Robardet, and Jean-François Boulicaut. Trend mining in dynamic attributed graphs. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I*, pages 654–669, 2013. (cited on page: 14).

[65] Trong Dinh Thac Do, Alexandre Termier, Anne Laurent, Benjamin Négrevergne, Behrooz Omidvar-Tehrani, and Sihem Amer-Yahia. PGLCM: efficient parallel mining of closed frequent gradual itemsets. *Knowl. Inf. Syst.*, 43(3):497–527, 2015. (cited on pages: 2, 3).

[66] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999*, pages 43–52, 1999. (cited on pages: 3, 21).

[67] X. Dong, D. Mavroeidis, F. Calabrese, and P. Frossard. Multiscale event detection in social media. *Dami*, 29(5):1374–1405, 2015. (cited on pages: 87, 88, and 101).

[68] Wouter Duivesteijn and Julia Thaele. Understanding where your classifier does (not) work - the scape model class for EMM. In *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*, pages 809–814, 2014. (cited on page: 22).

[69] Wouter Duivesteijn, Arno J. Knobbe, Ad Feelders, and Matthijs van Leeuwen. Subgroup discovery meets bayesian networks – an exceptional model mining approach. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 158–167, 2010. (cited on page: 22).

[70] Wouter Duivesteijn, Ad Feelders, and Arno J. Knobbe. Different slopes for different folks: mining for exceptional regression models with cook's distance. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 868–876, 2012. (cited on pages: 22, 23).

[71] Wouter Duivesteijn, Ad Feelders, and Arno J. Knobbe. Exceptional model mining - supervised descriptive local pattern mining with complex target concepts. *Data Min. Knowl. Discov.*, 30(1): 47–98, 2016. (cited on pages: 6, 10, 21, 22, 32, and 137).

[72] V. Dzyuba and M. van Leeuwen. Interactive discovery of interesting subgroup sets. In *IDA*, pages 150–161, 2013. (cited on pages: 3, 6, 11, and 27).

[73] Vladimir Dzyuba, Matthijs van Leeuwen, Siegfried Nijssen, and Luc De Raedt. Interactive learning of pattern rankings. *International Journal on Artificial Intelligence Tools*, 23(6), 2014. (cited on pages: 6, 11, 27, 28, 29, and 84).

[74] David Eppstein and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. In *International Symposium on Experimental Algorithms*, pages 364–375. Springer, 2011. (cited on page: 69).

[75] David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*, pages 403–414, 2010. (cited on page: 12).

[76] Martin Ester, Rong Ge, Byron J. Gao, Zengjian Hu, and Boaz Ben-Moshe. Joint cluster analysis of attribute data and relationship data: the connected k-center problem. In *Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, pages 246–257, 2006. (cited on page: 16).

[77] Géraud Le Falher, Aristides Gionis, and Michael Mathioudakis. Where is the soho of rome? measures and algorithms for finding similar neighborhoods in cities. In *Proceedings of the Ninth International Conference on Web and Social Media, ICWSM 2015, University of Oxford, Oxford, UK, May 26-29, 2015*, pages 228–237, 2015. (cited on pages: 5, 43, and 129).

[78] Yixiang Fang, Reynold Cheng, Yun Li, Xiaojun Chen, Jie Zhang, et al. Exploring communities in large profiled graphs. *IEEE Transactions on Knowledge & Data Engineering*, (1):1–1. (cited on page: 20).

[79] Yixiang Fang, Reynold Cheng, Siqiang Luo, and Jiafeng Hu. Effective community search for large attributed graphs. *PVLDB*, 9(12):1233–1244, 2016.

[80] Yixiang Fang, Reynold Cheng, Yankai Chen, Siqiang Luo, and Jiafeng Hu. Effective and efficient attributed community search. *VLDB J.*, 26(6):803–828, 2017.

[81] Yixiang Fang, Reynold Cheng, Xiaodong Li, Siqiang Luo, and Jiafeng Hu. Effective community search over large spatial graphs. *PVLDB*, 10(6):709–720, 2017. (cited on page: 20).

[82] Yixiang Fang, Zheng Wang, Reynold Cheng, Xiaodong Li, Siqiang Luo, Jiafeng Hu, and Xiaojun Chen. On spatial-aware community search. *IEEE Transactions on Knowledge and Data Engineering*, 2018. (cited on page: 20).

[83] Johannes Fischer, Volker Heun, and Stefan Kramer. Optimal string mining under frequency constraints. In *Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18-22, 2006, Proceedings*, pages 139–150, 2006. (cited on page: 2).

[84] Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 150–160, 2000. (cited on page: 10).

[85] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010. (cited on page: 10).

[86] Mutsumi Fukuzaki, Mio Seki, Hisashi Kashima, and Jun Sese. Finding itemset-sharing patterns in a large itemset-associated graph. In *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II*, pages 147–159, 2010. (cited on pages: 5, 10, 18, and 32).

[87] Dragan Gamberger and Nada Lavrac. Expert-guided subgroup discovery: Methodology and application. *J. Artif. Intell. Res.*, 17:501–527, 2002. (cited on pages: 4, 22).

[88] Dario Garcia-Gasulla, Sergio Álvarez-Napagao, Arturo Tejeda-Gómez, Luis Oliva Felipe, Ignasi Gómez-Sebastià, Javier Béjar, and Javier Vázquez-Salceda. Social network data analysis for event detection. In *ECAI*, pages 1009–1010, 2014. (cited on page: 88).

[89] Alain Gély, Lhouari Nourine, and Bachir Sadi. Enumeration aspects of maximal cliques and bicliques. *Discrete Applied Mathematics*, 157(7):1447–1459, 2009. (cited on pages: 2, 10, and 12).

[90] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 330–339, 2007. (cited on page: 2).

[91] David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 721–732, 2005. (cited on page: 10).

[92] Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas. Assessing data mining results via swap randomization. *TKDD*, 1(3):14, 2007. (cited on pages: 3, 26).

[93] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002. (cited on page: 10).

[94] Karam Gouda and Mohammed Javeed Zaki. Efficiently mining maximal frequent itemsets. In *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 163–170, 2001. (cited on page: 3).

[95] Andrei Grigoriev. A relationship between gene expression and protein interactions on the proteome scale: analysis of the bacteriophage t7 and the yeast saccharomyces cerevisiae. *Nucleic acids research*, 29(17):3513–3519, 2001. (cited on page: 10).

[96] Mael Gueguen, Olivier Sentieys, and Alexandre Termier. Accelerating itemset sampling using satisfiability constraints on FPGA. In *Design, Automation & Test in Europe Conference &*

Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019, pages 1046–1051, 2019. (cited on page: 4).

[97] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5): 93:1–93:42, 2019. (cited on page: 141).

[98] Stephan Günnemann, Ines Färber, Brigitte Boden, and Thomas Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 845–850, 2010. (cited on pages: 5, 10, 13, 17, 32, 44, 74, and 76).

[99] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. DB-CSC: A density-based approach for subspace clustering in graphs with feature vectors. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011. Proceedings, Part I*, pages 565–580, 2011. (cited on page: 17).

[100] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. Finding density-based subspace clusters in graphs with feature vectors. *Data Min. Knowl. Discov.*, 25(2):243–269, 2012. (cited on page: 17).

[101] Stephan Günnemann, Ines Färber, Sebastian Raubach, and Thomas Seidl. Spectral subspace clustering for graphs with feature vectors. In *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, pages 231–240, 2013. (cited on page: 17).

[102] Jiawei Han, Jianyong Wang, Ying Lu, and Petre Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 211–218, 2002. (cited on page: 27).

[103] Kai Han, Fei Gui, Xiaokui Xiao, Jing Tang, Yuntian He, Zongmai Cao, and He Huang. Efficient and effective algorithms for clustering uncertain graphs. *PVLDB*, 12(6):667–680, 2019. (cited on page: 141).

[104] Daniel Hanisch, Alexander Zien, Ralf Zimmer, and Thomas Lengauer. Co-clustering of biological networks and gene expression data. In *Proceedings of the Tenth International Conference on Intelligent Systems for Molecular Biology, August 3-7, 2002, Edmonton, Alberta, Canada*, pages 145–154, 2002. (cited on page: 10).

[105] Peter Harremoës. Binomial and poisson distributions as maximum entropy distributions. *IEEE Trans. Information Theory*, 47(5):2039–2041, 2001. (cited on page: 122).

[106] Mohammad Al Hasan and Mohammed J. Zaki. Output space sampling for graph patterns. *PVLDB*, 2(1):730–741, 2009. (cited on page: 32).

[107] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *SIGIR*, pages 207–214. ACM, 2007. (cited on pages: 88, 89).

[108] Pierre Houdyer, Albrecht Zimmermann, Mehdi Kaytoue, Marc Plantevit, Joseph Mitchell, and Céline Robardet. Gazouille: Detecting and illustrating local events from geolocalized social media streams. In *ECML PKDD 2015*, pages 276–280, 2015. (cited on page: 88).

[109] Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. In *Proceedings Thirteenth International Conference on Intelligent Systems for Molecular Biology 2005, Detroit, MI, USA, 25-29 June 2005*, pages 213–221, 2005. (cited on page: 10).

[110] Ting Hua, Feng Chen, Liang Zhao, Chang-Tien Lu, and Naren Ramakrishnan. STED: semi-supervised targeted-interest event detectionin in twitter. In *ACM SIGKDD*, pages 1466–1469, 2013. (cited on page: 87).

[111] Ting Hua, Feng Chen, Liang Zhao, Chang-Tien Lu, and Naren Ramakrishnan. Automatic targeted-domain spatiotemporal event detection in twitter. *GeoInformatica*, 20(4):765–795, 2016. (cited on page: 87).

[112] Jun Huan, Wei Wang, Jan F. Prins, and Jiong Yang. SPIN: mining maximal frequent subgraphs from graph databases. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 581–586, 2004. (cited on page: 3).

[113] Xin Huang and Laks V. S. Lakshmanan. Attribute-driven community search. *PVLDB*, 10(9): 949–960, 2017. (cited on page: 20).

[114] Xin Huang, Laks V. S. Lakshmanan, and Jianliang Xu. Community search over big graphs: Models, algorithms, and opportunities. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 1451–1454, 2017. (cited on page: 20).

[115] G. Ifrim, B. Shi, and I. Brigadir. Event detection in twitter using aggressive filtering and hierarchical tweet clustering. In *snow@WWW*, pages 33–40, 2014. (cited on pages: 88, 101).

[116] Robert T Jensen and Nolan H Miller. Giffen behavior and subsistence consumption. *American Economic Review*, 98(4):1553–77, 2008. (cited on page: 23).

[117] Chuntao Jiang, Frans Coenen, and Michele Zito. A survey of frequent subgraph mining algorithms. *Knowledge Eng. Review*, 28(1):75–105, 2013. (cited on page: 10).

[118] Bo Kang, Jefrey Lijffijt, Raúl Santos-Rodríguez, and Tijl De Bie. SICA: subjectively interesting component analysis. *Data Min. Knowl. Discov.*, 32(4):949–987, 2018. (cited on page: 25).

[119] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972. (cited on page: 12).

[120] Mehdi Kaytoue, Sergei O. Kuznetsov, and Amedeo Napoli. Revisiting numerical pattern mining with formal concept analysis. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1342–1347, 2011. (cited on page: 2).

[121] Mehdi Kaytoue, Marc Plantevit, Albrecht Zimmermann, Ahmed Anes Bendimerad, and Céline Robardet. Exceptional contextual subgraph mining. *Machine Learning*, 106(8):1171–1211, 2017. (cited on pages: 11, 14, 23, and 44).

[122] Arijit Khan, Xifeng Yan, and Kun-Lung Wu. Towards proximity pattern mining in large graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 867–878, 2010. (cited on pages: 5, 10, 17, and 18).

[123] Martin Kirchgessner, Vincent Leroy, Alexandre Termier, Sihem Amer-Yahia, and Marie-Christine Rousset. Toppi: An efficient algorithm for item-centric mining. In *Big Data Analytics and Knowledge Discovery - 18th International Conference, DaWaK 2016, Porto, Portugal, September 6-8, 2016, Proceedings*, pages 19–33, 2016. (cited on page: 3).

[124] M. Klazar. Twelve countings with rooted plane trees. *European J. Combin.*, 18(2):195–210, 1997. (cited on page: 127).

[125] Willi Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. 1996. (cited on pages: 6, 21).

[126] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 313–320, 2001. (cited on pages: 2, 10).

[127] Sergei O. Kuznetsov. Learning of simple conceptual graphs from positive and negative examples. In *Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD '99, Prague, Czech Republic, September 15-18, 1999, Proceedings*, pages 384–391, 1999. (cited on page: 35).

[128] Nada Lavrac, Peter A. Flach, and Blaz Zupan. Rule evaluation measures: A unifying view. In *Inductive Logic Programming, 9th International Workshop, ILP-99, Bled, Slovenia, June 24-27, 1999, Proceedings*, pages 174–185, 1999. (cited on page: 21).

[129] Nada Lavrac, Bojan Cestnik, Dragan Gamberger, and Peter A. Flach. Decision support through subgroup discovery: Three case studies and the lessons learned. *Machine Learning*, 57(1-2): 115–143, 2004. (cited on pages: 4, 22).

[130] Nada Lavrac, Branko Kavsek, Peter A. Flach, and Ljupco Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004. (cited on pages: 3, 6, 10, 21, 32, 34, 35, 131, and 137).

[131] Nada Lavrac, Anze Vavpetic, Larisa N. Soldatova, Igor Trajkovski, and Petra Kralj Novak. Using ontologies in semantic data mining with SEGS and g-segs. In *Discovery Science*, pages 165–178, 2011. (cited on page: 116).

[132] Sau Dan Lee and Luc De Raedt. An efficient algorithm for mining string databases under constraints. In *KDID 2004, Knowledge Discovery in Inductive Databases, Proceedings of the Third International Workshop on Knowledge Discovery inInductive Databases, Pisa, Italy, September 20, 2004, Revised Selected and Invited Papers*, pages 108–129, 2004. (cited on page: 2).

[133] Dennis Leman, Ad Feelders, and Arno J. Knobbe. Exceptional model mining. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II*, pages 1–16, 2008. (cited on pages: 21, 22).

[134] Florian Lemmerich, Mathias Rohlfs, and Martin Atzmüller. Fast discovery of relevant subgroup patterns. In *Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference, May 19-21, 2010, Daytona Beach, Florida, USA*, 2010. (cited on page: 22).

[135] Florian Lemmerich, Martin Becker, Philipp Singer, Denis Helic, Andreas Hotho, and Markus Strohmaier. Mining subgroups with exceptional transition behavior. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 965–974, 2016. (cited on pages: 24, 44).

[136] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. Radar: Residual analysis for anomaly detection in attributed networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2152–2158, 2017. (cited on pages: 19, 20).

[137] Nan Li, Huan Sun, Kyle C. Chipman, Jemin George, and Xifeng Yan. A probabilistic approach to uncovering attributed graph anomalies. In *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 82–90, 2014. (cited on page: 19).

[138] Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang. Tedas: A twitter-based event detection and analysis system. In *ICDE'12*, pages 1273–1276, 2012. ISBN 978-0-7695-4747-3. (cited on page: 87).

[139] Yafang Li, Xiangnan Kong, Caiyan Jia, and Jianqiang Li. Clustering uncertain graphs with node attributes. In *Proceedings of The 10th Asian Conference on Machine Learning, ACML 2018, Beijing, China, November 14-16, 2018.*, pages 232–247, 2018. (cited on page: 141).

[140] Jefrey Lijffijt, Panagiotis Papapetrou, and Kai Puolamʻaki. A statistical significance testing approach to mining the most informative set of patterns. *DMKD*, 28(1):238–263, 2014. (cited on page: 61).

[141] Jefrey Lijffijt, Eirini Spyropoulou, Bo Kang, and Tijl De Bie. P-n-rminer: a generic framework for mining interesting structured relational patterns. *I. J. Data Science and Analytics*, 1(1):61–76, 2016. (cited on pages: 25, 74, and 75).

[142] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York City, New York, USA, August 27-31, 1998*, pages 80–86, 1998. (cited on page: 21).

[143] Guimei Liu and Limsoon Wong. Effective pruning techniques for mining quasi-cliques. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II*, pages 33–49, 2008. (cited on pages: 2, 10, 12, and 13).

[144] Haishan Liu. Towards semantic data mining. In *ISWC*, pages 7–11, 2010. (cited on page: 116).

[145] Felipe Llinares-López, Mahito Sugiyama, Laetitia Papaxanthos, and Karsten M. Borgwardt. Fast and memory-efficient significant pattern mining via permutation testing. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 725–734, 2015. (cited on page: 3).

[146] Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *Algorithm Theory - SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, Humlebaek, Denmark, July 8-10, 2004, Proceedings*, pages 260–272, 2004. (cited on pages: 2, 10).

[147] Heikki Mannila and Hannu Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 146–151, 1996. (cited on page: 2).

[148] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258, 1997. (cited on page: 1).

[149] Hideo Matsuda, T. Ishihara, and Akihiro Hashimoto. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theor. Comput. Sci.*, 210(2):305–325, 1999. (cited on page: 13).

[150] Nicolas Méger and Christophe Rigotti. Constraint-based mining of episode rules and optimal window sizes. In *Knowledge Discovery in Databases: PKDD 2004, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, September 20-24, 2004, Proceedings*, pages 313–324, 2004. (cited on page: 2).

[151] Benjamin A. Miller, Michelle S. Beard, Patrick J. Wolfe, and Nadya T. Bliss. A spectral framework for anomalous subgraph detection. *IEEE Trans. Signal Processing*, 63(16):4191–4206, 2015. (cited on page: 24).

[152] Alan Mislove, Massimiliano Marcon, P. Krishna Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC 2007, San Diego, California, USA, October 24-26, 2007*, pages 29–42, 2007. (cited on page: 10).

[153] Sandy Moens and Mario Boley. Instant exceptional model mining using weighted controlled pattern sampling. In *Advances in Intelligent Data Analysis XIII - 13th International Symposium, IDA 2014, Leuven, Belgium, October 30 - November 1, 2014. Proceedings*, pages 203–214, 2014. (cited on pages: 4, 22).

[154] Sandy Moens and Bart Goethals. Randomly sampling maximal itemsets. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, IDEA@KDD 2013, Chicago, Illinois, USA, August 11, 2013*, pages 79–86, 2013. (cited on page: 4).

[155] Maëlle Moranges, Marc Plantevit, Arnaud P. Fournel, Moustafa Bensafi, and Céline Robardet. Exceptional attributed subgraph mining to understand the olfactory percept. In *Discovery Science - 21st International Conference, DS 2018, Limassol, Cyprus, October 29-31, 2018, Proceedings*, pages 276–291, 2018. (cited on page: 32).

[156] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining cohesive patterns from graphs with feature vectors. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 593–604, 2009. (cited on pages: 5, 10, 17, 32, and 44).

[157] Pierre-Nicolas Mougel, Christophe Rigotti, and Olivier Gandrillon. Finding collections of k-clique percolated components in attributed graphs. In *Advances in Knowledge Discovery and Data Mining - 16th Pacific-Asia Conference, PAKDD 2012, Kuala Lumpur, Malaysia, May 29 - June 1, 2012, Proceedings, Part II*, pages 181–192, 2012. (cited on page: 19).

[158] Pierre-Nicolas Mougel, Christophe Rigotti, Marc Plantevit, and Olivier Gandrillon. Finding maximal homogeneous clique sets. *Knowl. Inf. Syst.*, 39(3):579–608, 2014. (cited on pages: 19, 44).

[159] Emmanuel Müller, Patricia Iglesias Sánchez, Yvonne Mülle, and Klemens Böhm. Ranking outlier nodes in subspaces of attributed graphs. In *Workshops Proceedings of the 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 216–222, 2013. (cited on pages: 13, 19, and 20).

[160] Benjamin Négrevergne, Alexandre Termier, Marie-Christine Rousset, and Jean-François Méhaut. Para miner: a generic pattern mining algorithm for multi-core architectures. *Data Min. Knowl. Discov.*, 28(3):593–633, 2014. (cited on page: 3).

[161] Jennifer Neville, Micah Adler, and David Jensen. Clustering relational data using attribute and link information. In *Proceedings of the text mining and link analysis workshop, 18th international joint conference on artificial intelligence*, pages 9–15. San Francisco, CA: Morgan Kaufmann Publishers, 2003. (cited on page: 16).

[162] Siegfried Nijssen. Tree mining. In *Encyclopedia of Machine Learning and Data Mining*, pages 1284–1292. 2017. (cited on page: 2).

[163] Siegfried Nijssen and Joost N. Kok. Frequent graph mining and its application to molecular databases. In *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: The Hague, Netherlands, 10-13 October 2004*, pages 4571–4577, 2004. (cited on page: 2).

[164] Petra Kralj Novak, Nada Lavrac, and Geoffrey I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403, 2009. (cited on page: 21).

[165] Victoria Pachón, Jacinto Mata Vázquez, Juan Luis Domínguez, and Manuel J. Maña López. Multi-objective evolutionary approach for subgroup discovery. In *Hybrid Artificial Intelligent Systems - 6th International Conference, HAIS 2011, Wroclaw, Poland, May 23-25, 2011, Proceedings, Part II*, pages 271–278, 2011. (cited on pages: 4, 22).

[166] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *WWW*, pages 161–172, 1998. (cited on page: 92).

[167] S. Papadopoulos, D. Corney, and L. Maria Aiello, editors. *SNOW Data Challenge*, volume 1150, 2014. (cited on page: 101).

[168] Jian Pei, Jiawei Han, and Laks V. S. Lakshmanan. Pushing convertible constraints in frequent itemset mining. *Data Min. Knowl. Discov.*, 8(3):227–252, 2004. (cited on page: 3).

[169] Jian Pei, Daxin Jiang, and Aidong Zhang. On mining cross-graph quasi-cliques. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 228–238, 2005. (cited on pages: 2, 10, 12, and 13).

[170] Leonardo Pellegrina and Fabio Vandin. Efficient mining of the most significant patterns with permutation testing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2070–2079, 2018. (cited on page: 27).

[171] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, and Qinghua Zheng. ANOMALOUS: A joint modeling approach for anomaly detection on attributed networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 3513–3519, 2018. (cited on pages: 19, 20).

[172] José Alberto Pérez-Melián, J. Alberto Conejero, and Cèsar Ferri Ramirez. Zipf's and benford's laws in twitter hashtags. In *EACL*, pages 84–93, 2017. (cited on page: 102).

[173] Charles E Perkins et al. *Ad hoc networking*, volume 1. Addison-wesley Reading, 2001. (cited on page: 10).

[174] Bryan Perozzi and Leman Akoglu. Scalable anomaly ranking of attributed neighborhoods. In *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016*, pages 207–215, 2016. (cited on page: 19).

[175] Adriana Prado, Marc Plantevit, Céline Robardet, and Jean-François Boulicaut. Mining graph topological patterns: Finding covariations among vertex descriptors. *IEEE Trans. Knowl. Data Eng.*, 25(9):2090–2104, 2013. (cited on pages: 19, 44).

[176] Guo-Jun Qi, Charu C. Aggarwal, Qi Tian, Heng Ji, and Thomas S. Huang. Exploring context and content links in social media: A latent space method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34 (5):850–862, 2012. (cited on page: 14).

[177] Luc De Raedt, Manfred Jaeger, Sau Dan Lee, and Heikki Mannila. A theory of inductive query answering. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 123–130, 2002. (cited on page: 2).

[178] Céline Robardet. Constraint-based pattern mining in dynamic graphs. In *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 950–955, 2009. (cited on page: 14).

[179] Daniel Rodríguez, Roberto Ruiz, José C. Riquelme, and Jesús S. Aguilar-Ruiz. Searching for rules to detect defective modules: A subgroup discovery approach. *Inf. Sci.*, 191:14–30, 2012. (cited on pages: 4, 22).

[180] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. Efficient community detection in large networks using content and links. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 1089–1098, 2013. (cited on page: 16).

[181] S. Rüping. Ranking interesting subgroups. In *ICML*, pages 913–920, 2009. (cited on pages: 27, 28).

[182] Heungmo Ryang and Unil Yun. Top-k high utility pattern mining with effective threshold raising strategies. *Knowl.-Based Syst.*, 76:109–126, 2015. (cited on page: 27).

[183] Tanay Kumar Saha and Mohammad Al Hasan. Fs^3: A sampling based method for top-k frequent subgraph mining. *Statistical Analysis and Data Mining*, 8(4):245–261, 2015. (cited on page: 32).

[184] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*, pages 851–860, 2010. (cited on page: 87).

[185] Patricia Iglesias Sánchez, Emmanuel Müller, Oretta Irmler, and Klemens Böhm. Local context selection for outlier ranking in graphs with multiple numeric node attributes. In *Conference on Scientific and Statistical Database Management, SSDBM '14, Aalborg, Denmark, June 30 - July 02, 2014*, pages 16:1–16:12, 2014. (cited on pages: 19, 20).

[186] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Systems, Man, and Cybernetics*, 13(3):353–362, 1983. (cited on page: 10).

[187] Jun Sese, Mio Seki, and Mutsumi Fukuzaki. Mining networks with shared items. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 1681–1684, 2010. (cited on pages: 18, 32).

[188] Jingwen Shang, Chaokun Wang, Changping Wang, Gaoyang Guo, and Jun Qian. AGAR: an attribute-based graph refining method for community search. In *Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory, EDB 2016, Jeju Island, Republic of Korea, October 17-19, 2016*, pages 65–66, 2016. (cited on page: 20).

[189] Abraham Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada, August 20-21, 1995*, pages 275–281, 1995. (cited on pages: 6, 11, and 24).

[190] 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., USA, July 24-25, 2010*, pages 119–126, 2010. (cited on pages: 5, 10, 18, and 32).

[191] Arlei Silva, Wagner Meira Jr., and Mohammed J. Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *PVLDB*, 5(5):466–477, 2012. (cited on pages: 18, 32, and 44).

[192] Arlei Silva, Petko Bogdanov, and Ambuj K. Singh. Hierarchical in-network attribute compression via importance sampling. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 951–962, 2015. (cited on page: 19).

[193] David M. Smith. Algorithm 814: Fortran 90 software for floating-point multiple precision arithmetic, gamma and related functions. *ACM Trans. Math. Softw.*, pages 377–387, 2001. (cited on page: 127).

[194] Laura M. Smith, Linhong Zhu, Kristina Lerman, and Allon G. Percus. Partitioning networks with node attributes by compressing information flow. *TKDD*, 11(2):15:1–15:26, 2016. (cited on page: 16).

[195] Arnaud Soulet. Two decades of pattern mining: Principles and methods. In *Business Intelligence - 6th European Summer School, eBISS 2016, Tours, France, July 3-8, 2016, Tutorial Lectures*, pages 59–78, 2016. (cited on page: 2).

[196] Arnaud Soulet, Chedy Raïssi, Marc Plantevit, and Bruno Crémilleux. Mining dominant patterns in the sky. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 655–664, 2011. (cited on page: 27).

[197] Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 939–948, 2010. (cited on page: 20).

[198] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, pages 3–17, 1996. (cited on page: 2).

[199] Aika Terada, Mariko Okada-Hatakeyama, Koji Tsuda, and Jun Sese. Statistical significance of combinatorial regulations. *Proceedings of the National Academy of Sciences*, 110(32):12996–13001, 2013. (cited on page: 21).

[200] Alexandre Termier. Distributed computing for enumeration. In *Encyclopedia of Algorithms*, pages 574–577. 2016. (cited on page: 3).

[201] Alexandre Termier, Marie-Christine Rousset, and Michèle Sebag. DRYADE: A new approach for discovering closed frequent trees in heterogeneous tree databases. In *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, pages 543–546, 2004. (cited on page: 2).

[202] Alexandre Termier, Marie-Christine Rousset, Michèle Sebag, Kouzou Ohara, Takashi Washio, and Hiroshi Motoda. Efficient mining of high branching factor attribute trees. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 785–788, 2005.

[203] Alexandre Termier, Marie-Christine Rousset, Michèle Sebag, Kouzou Ohara, Takashi Washio, and Hiroshi Motoda. Dryadeparent, an efficient and robust closed attribute tree mining algorithm. *IEEE Trans. Knowl. Data Eng.*, 20(3):300–320, 2008. (cited on page: 2).

[204] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.*, 363(1): 28–42, 2006. (cited on page: 12).

[205] Julian R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976. (cited on page: 10).

[206] Takeaki Uno. An efficient algorithm for solving pseudo clique enumeration problem. *Algorithmica*, 56(1):3–16, 2010. (cited on pages: 3, 10, and 13).

[207] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. LCM: an efficient algorithm for enumerating frequent closed item sets. In *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, 2003. (cited on page: 3).

[208] Matthijs van Leeuwen, Tijl De Bie, Eirini Spyropoulou, and Cédric Mesnage. Subjective interestingness of subgraph patterns. *Machine Learning*, 105(1):41–75, 2016. (cited on pages: 4, 25).

[209] Anze Vavpetic and Nada Lavrac. Semantic subgroup discovery systems and workflows in the sdm-toolkit. *Comput. J.*, 56(3):304–320, 2013. (cited on page: 116).

[210] Anze Vavpetic, Petra Kralj Novak, Miha Grcar, Igor Mozetic, and Nada Lavrac. Semantic data mining of financial news articles. In *Discovery Science*, pages 294–307, 2013. (cited on page: 116).

[211] Maximilian Walther and Michael Kaisser. Geo-spatial event detection in the twitter stream. In *European Conference on Advances in IR*, pages 356–367, 2013. ISBN 978-3-642-36972-8. (cited on page: 88).

[212] Bing Wang, Lang Cao, Hideyuki Suzuki, and Kazuyuki Aihara. Epidemic spread in adaptive networks with multitype agents. *Journal of Physics A: Mathematical and Theoretical*, 44(3): 035101, 2010. (cited on page: 10).

[213] Jia Wang, James Cheng, and Ada Wai-Chee Fu. Redundancy-aware maximal cliques. In *KDD 2013*, pages 122–130, 2013. (cited on page: 90).

[214] Kazufumi Watanabe, Masanao Ochi, Makoto Okabe, and Rikio Onai. Jasmine: A real-time local-event detection system based on geolocation information propagated to microblogs. In *CIKM*, pages 2541–2544, 2011. (cited on page: 88).

[215] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440, 1998. (cited on page: 11).

[216] Geoffrey I. Webb. OPUS: an efficient admissible algorithm for unordered search. *J. Artif. Intell. Res.*, 3:431–465, 1995. (cited on page: 3).

[217] Yifang Wei, Lisa Singh, Brian Gallagher, and David Buttler. Overlapping target event and story line detection of online newspaper articles. In *DSAA 2016*, pages 222–232, 2016. (cited on page: 87).

[218] J. Weng and B.-S. Lee. Event detection in twitter. In *ICWSM*, 2011. (cited on pages: 88, 101).

[219] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *Principles of Data Mining and Knowledge Discovery, First European Symposium, PKDD '97, Trondheim, Norway, June 24-27, 1997, Proceedings*, pages 78–87, 1997. (cited on page: 21).

[220] D. Xin, X. Shen, Q. Mei, and J. Han. Discovering interesting patterns through user's interactive feedback. In *KDD*, pages 773–778, 2006. (cited on pages: 3, 6, 11, 27, and 28).

[221] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. GBAGC: A general bayesian framework for attributed graph clustering. *TKDD*, 9(1):5:1–5:43, 2014. (cited on page: 16).

[222] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 721–724, 2002. (cited on pages: 2, 10).

[223] Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large datasets. In *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*, pages 166–177, 2003. (cited on page: 3).

[224] Jaewon Yang, Julian J. McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, pages 1151–1156, 2013. (cited on pages: 10, 16).

[225] Mohammed Javeed Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 71–80, 2002. (cited on page: 2).

[226] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *Proceedings of the Twelfth ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006, pages 797–802, 2006. (cited on pages: 2, 10, 12, and 13).

[227] Chao Zhang, Guangyu Zhou, Quan Yuan, Honglei Zhuang, Yu Zheng, Lance M. Kaplan, Shaowen Wang, and Jiawei Han. Geoburst: Real-time local event detection in geo-tagged tweet streams. In *ACM SIGIR*, pages 513–522, 2016. (cited on pages: 87, 88, 89, and 101).

[228] Chao Zhang, Liyuan Liu, Dongming Lei, Quan Yuan, Honglei Zhuang, Tim Hanratty, and Jiawei Han. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *ACM SIGKDD*, pages 595–604, 2017. (cited on page: 88).

[229] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. When engagement meets similarity: Efficient (k, r)-core computation on social networks. *PVLDB*, 10(10):998–1009, 2017. (cited on page: 19).

[230] Yuan Zhang, Elizaveta Levina, and Ji Zhu. Community detection in networks with node features. *CoRR*, abs/1509.01173, 2015. (cited on page: 16).

[231] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009. (cited on page: 16).

[232] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn J. Keogh. Matrix profile XI: SCRIMP++: time series motif discovery at interactive speeds. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 837–846, 2018. (cited on pages: 4, 140).

[233] Zhaonian Zou. Polynomial-time algorithm for finding densest subgraphs in uncertain graphs. In *Proceedings of MLG Workshop*, 2013. (cited on page: 141).

[234] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Frequent subgraph pattern mining on uncertain graph data. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 583–592, 2009. (cited on page: 141).

[235] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Mining frequent subgraph patterns from uncertain graph data. *IEEE Trans. Knowl. Data Eng.*, 22(9):1203–1218, 2010. (cited on page: 141).

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : BENDIMERAD
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 05/09/2019

Prénoms : Ahmed Anes

TITRE : Mining useful patterns in attributed graphs

NATURE : Doctorat

Numéro d'ordre : 2019LYSEI058

Ecole doctorale : InfoMaths (ED 512)

Spécialité : Informatique

RESUME :
Un graphe est une structure qui permet de modéliser efficacement une large variété de données. Par exemple, un réseau social peut être représenté avec un graphe où les personnes sont les sommets, et leurs liens d'amitiés sont les arêtes. Ces graphes peuvent être augmentés avec des attributs décrivant ses sommets. Dans un réseau social, chaque personne peut être décrite par son âge, ses centres d'intérêts, etc. C'est ce qu'on appelle un graphe attribué. L'analyse de ce type de graphes peut offrir une grande opportunité pour extraire des informations utiles et actionnables. Cela permet d'identifier des communautés ayant des centres d'intérêts particuliers dans un réseau social, de détecter des évènements à partir des tweets partagés, etc. Dans cette thèse, nous adressons le problème de fouille de graphes attribués. Plus précisément, nous proposons des méthodes qui analysent un graphe pour identifier des motifs : des sous-graphes ayant des caractéristiques particulières. Bien que ce problème a intéressé un grand nombre de chercheurs depuis des années, il reste encore plusieurs défis à relever. Nous adressons les questions : quand est-ce qu'un motif est intéressant pour l'utilisateur ? plusieurs facteurs entrent en jeu. Nous considérons qu'un motif est intéressant : (1) s'il montre une exceptionnalité par rapport au reste du graphe, (2) s'il donne une nouvelle information à l'utilisateur (il est imprévu), (3) s'il communique une information qui fait partie du centre d'intérêt de l'utilisateur (préférences). Pour mesurer la qualité d'un motif, nous proposons des modèles qui prennent en compte un ou plusieurs de ces trois facteurs. Nous définissons des algorithmes qui déterminent les meilleurs motifs selon chaque modèle proposé, et nous effectuons des études empiriques pour évaluer l'efficacité de chacun de ces algorithmes.

MOTS-CLÉS : Fouille de graphes attribués, fouille de données centrée sur l'utilisateur, fouille de modèles exceptionnels, analyse des dynamiques urbaines, détection d'évènements sur les réseaux sociaux.

Laboratoire (s) de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)

Directeur de thèse:
Pr. Céline Robardet (INSA-Lyon),
Dr. Marc Plantevit (Université Claude Bernard Lyon 1).

Président de jury :

Composition du jury :
Pr. Aristides Gionis (Aalto University),
Pr. Marie-Christine Rousset (Université Grenoble Alpes),
Pr. Tijl De Bie (Ghent University),
Pr. Alexandre Termier (Université de Rennes 1),
Dr. Siegfried Nijssen (Université Catholique de Louvain),
Pr. Céline Robardet (INSA-Lyon),
Dr. Marc Plantevit (Université Claude Bernard Lyon 1).