

THÈSE

présentée devant

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

pour obtenir

LE GRADE DE DOCTEUR

Spécialité

INFORMATIQUE

École Doctorale : Informatique et Mathématiques

par

Pierre-Nicolas MOUGEL

FINDING HOMOGENEOUS COLLECTIONS OF
DENSE SUBGRAPHS USING CONSTRAINT-BASED
DATA MINING APPROACHES

APPLICATION TO THE ANALYSIS OF SCIENTIFIC COLLABORATION
NETWORKS AND PROTEIN INTERACTION GRAPHS

Soutenance le 14 Septembre 2012 devant le jury :

Jean-François BOULICAUT	INSA de Lyon	Examineur
Bruno CRÉMILLEUX	Université de Caen	Examineur
Florent MASSEGLIA	INRIA Sophia Antipolis	Rapporteur
Dino PEDRESCHI	Università di Pisa, IT	Rapporteur
Pascal PONCELET	Université de Montpellier 2	Examineur
Christophe RIGOTTI	INSA de Lyon	Directeur
Céline ROUVEIROL	Université Paris-Nord	Examineur

Pierre-Nicolas Mougel: *Finding Homogeneous Collections of Dense Subgraphs Using Constraint-Based Data Mining Approaches*, PhD Thesis, © October 2009–September 2012

SUPERVISOR:
Christophe Rigotti

TIME FRAME:
October 2009–September 2012

ABSTRACT

The work presented in this thesis deals with data mining approaches for the analysis of attributed graphs. An attributed graph is a graph where properties, encoded by means of attributes, are associated to each vertex. In such data, our objective is the discovery of subgraphs formed by several dense groups of vertices that are homogeneous with respect to the attributes.

More precisely, we define the constraint-based extraction of collections of subgraphs densely connected and such that the vertices share enough attributes. To this aim, we propose two new classes of patterns along with sound and complete algorithms to compute them efficiently using constraint-based approaches. The first family of patterns, named Maximal Homogeneous Clique Set (MHCS), contains patterns satisfying constraints on the number of dense subgraphs, on the size of these subgraphs, and on the number of shared attributes. The second class of patterns, named Collection of Homogeneous k -clique Percolated components (CoHoP), is based on a relaxed notion of density in order to handle missing values.

Both approaches are used for the analysis of scientific collaboration networks and protein-protein interaction networks. The extracted patterns exhibit structures useful in a decision support process. Indeed, in a scientific collaboration network, the analysis of such structures might give hints to propose new collaborations between researchers working on the same subjects. In a protein-protein interaction network, the analysis of the extracted patterns can be used to study the relationships between modules of proteins involved in similar biological situations. The analysis of the performances, on real and synthetic data, with respect to different attributed graph characteristics, shows that the proposed approaches scale well for large datasets.

RÉSUMÉ

Ce travail de thèse concerne la fouille de données sur des graphes attribués. Il s'agit de graphes dans lesquels des propriétés, encodées sous forme d'attributs, sont associées à chaque sommet. Notre objectif est la découverte, dans ce type de données, de sous-graphes organisés en plusieurs groupes de sommets fortement connectés et homogènes au regard des attributs.

Plus précisément, nous définissons l'extraction sous contraintes d'ensembles de sous-graphes densément connectés et tels que les sommets partagent suffisamment d'attributs. Pour cela nous proposons deux familles de motifs originales ainsi que les algorithmes justes et complets permettant leur extraction efficace sous contraintes. La première famille, nommée Ensembles Maximaux de Cliques Homogènes, correspond à des motifs satisfaisant des contraintes concernant le nombre de sous-graphes denses, la taille de ces sous-graphes et le nombre d'attributs partagés. La seconde famille, nommée Collections Homogènes de k -cliques Percolées emploie quant à elle une notion de densité plus relaxée permettant d'adapter la méthode aux données avec des valeurs manquantes.

Ces deux méthodes sont appliquées à l'analyse de deux types de réseaux, les réseaux de coopérations entre chercheurs et les réseaux d'interactions de protéines. Les motifs obtenus mettent en évidence des structures utiles dans un processus de prise de décision. Ainsi, dans un réseau de coopérations entre chercheurs, l'analyse de ces structures peut aider à la mise en place de collaborations scientifiques entre des groupes travaillant sur un même domaine. Dans le contexte d'un graphe de protéines, les structures exhibées permettent d'étudier les relations entre des modules de protéines intervenant dans des situations biologiques similaires. L'étude des performances en fonction de différentes caractéristiques de graphes attribués réels et synthétiques montre que les approches proposées sont utilisables sur de grands jeux de données.

PUBLICATIONS

Most of the ideas presented in this thesis appeared previously in the following publications:

INTERNATIONAL CONFERENCES

- [CMB09] Loïc Cerf, Pierre-Nicolas Mougel, and Jean-Francois Boulicaut. Agglomerating Local Patterns Hierarchically with ALPHA. In *Proc. of the ACM Int. Conf. on Information and Knowledge Management (CIKM)*, pages 1753–1756, November 2009.
- [MPR⁺10] Pierre-Nicolas Mougel, Marc Plantevit, Christophe Rigotti, Olivier Gandrillon, and Jean-Francois Boulicaut. Constraint-Based Mining of Sets of Cliques Sharing Vertex Properties. In *Proc. of the Workshop on Analysis of Complex NEtworks (ACNE) co-located with ECML PKDD*, pages 48–62, September 2010.
- [MRG12a] Pierre-Nicolas Mougel, Christophe Rigotti, and Olivier Gandrillon. Finding Collections of k-Clique Percolated Components in Attributed Graphs. In *Proc. of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 181–192, April 2012.
- [MRG12b] Pierre-Nicolas Mougel, Christophe Rigotti, and Olivier Gandrillon. Finding Collections of Protein Modules in Protein-Protein Interaction Networks . In *Proc. of Bioinformatics and Computational Biology (BICoB)*, March 2012.

NATIONAL CONFERENCES

- [MPR⁺11] Pierre-Nicolas Mougel, Marc Plantevit, Christophe Rigotti, Olivier Gandrillon, and Jean-Francois Boulicaut. Extraction sous Contraintes d’Ensembles de Cliques Homogènes. In *Proc. of Conférence Francophone sur l’Extraction et la Gestion des Connaissances (EGC)*, pages 443–454, January 2011.

POSTERS

- [MPR⁺10] Pierre-Nicolas Mougel, Marc Plantevit, Christophe Rigotti, Olivier Gandrillon, and Jean-Francois Boulicaut. A Data Mining Approach to Highlight Relations Between Functional Modules. In *Integrative Post-Genomics (IPG)*, November 2010.

ACKNOWLEDGMENTS

J'aurais aimé remercier toutes les personnes que j'ai pu rencontrer et avec qui échanger a été si agréable. Même si je n'ai pas la place pour remercier tout le monde, je souhaiterais commencer avec des amis qui me sont très chers. Tout d'abord à Lolie, pour tous les meilleurs moments. Merci également à Dora et Vince avec qui j'espère avoir encore longtemps l'occasion de passer des soirées. Merci aussi Ben pour un nombre de raisons qui ajouterait sûrement quelques pages à ce manuscrit. Merci à Arnaud et Simon, les amis de l'INSA, avec qui il est très agréable de discuter recherche ou d'autres sujets.

Merci aussi à tous les doctorants avec qui j'ai partagé le bureau 501.319 pour les discussions, pas toujours scientifiques, mais à chaque fois agréables, Loïc, Ngan, Stephan, Bérénice et Élise. Merci à tous les membres des équipes Turing / Combining / actuellement DM2L et Beagle pour leurs conseils et les discussions, et tout particulièrement à Jean-François sans qui je n'aurais peut-être pas pu faire cette thèse ainsi que pour sa direction scientifique exceptionnelle de l'équipe.

Cette thèse n'aurait pas été possible sans Christophe, que je remercie non seulement pour son apport et ses qualités scientifiques, mais également pour ses nombreux conseils dans des domaines pas uniquement scientifiques et le temps passé à me les offrir. Merci aussi pour cette bonne humeur continuelle qui a permis de rendre ces trois années très agréables.

Mes remerciements vont aussi à mes rapporteurs, Dino Pedreschi et Florent Masegla pour leur temps et leurs remarques ainsi qu'aux autres membres du jury, Céline Rouveirol, Jean-François Boulicaut, Bruno Crémilleux, et Pascal Poncelet pour avoir bien voulu participer. Merci également aux membres des projets ANR Bingo2 (ANR-07-MDCO-014) et REHMI (financé par l'institut rhône-alpin des systèmes complexes) et en particulier à Olivier Gandrillon pour son expertise précieuse en biologie. Merci également aux membres du projet Foster (ANR-2010-COSI-012-02) avec qui j'aurai le plaisir de continuer mes travaux de recherche en Post-Doctorat.

Enfin, merci aux personnes avec qui j'ai passé un bout de ma vie. Sandie, qui a dû supporter mon côté geek et de longs monologues pendant la rédaction du manuscrit. Mes frères et ma choupinette pour des longs monologues un peu plus anciens, mon père qui m'a transmis sa passion de l'informatique et ma maman qui a permis que tout cela se réalise.

CONTENTS

I INTRODUCTION	1
Context	3
Problem	5
Contribution	6
Organization of the thesis	6
II STATE OF THE ART	9
OUTLINE	11
1 LOCAL PATTERN MINING IN BINARY RELATIONS	13
1.1 The binary relation context	13
1.2 Local patterns in binary relations	14
1.2.1 Frequent itemsets	14
1.2.2 Frequent closed itemsets	16
1.2.3 Frequent closed error-tolerant itemsets	16
1.2.4 Constraints on patterns and pattern sets	17
2 LOCAL PATTERN MINING IN GRAPHS	19
2.1 The graph context	19
2.1.1 Graph measures	20
2.2 Local patterns in graphs	20
2.2.1 Cliques and maximal cliques	21
2.2.2 Quasi-cliques	22
2.2.3 k-clique percolated components	23
3 MINING ATTRIBUTED GRAPHS	27
3.1 The attributed graph context	27
3.2 Global approaches in attributed graphs	28
3.3 Local patterns in attributed graphs	30
3.3.1 Application specific approaches	30
3.3.2 General frameworks	30
CONCLUSION	33
III A PATTERN AS A COLLECTION OF SUBGRAPHS	35
INTRODUCTION	37
4 MINING COLLECTIONS OF CLIQUES	41
4.1 Pattern definition	41
4.1.1 The homogeneity constraint: C_{α}^{homo}	42
4.1.2 The topology constraint: $C_{\gamma,k}^{\text{clique}}$	42
4.1.3 The separation constraint: C^{sep}	43
4.1.4 Reducing the collection of patterns	43
4.2 Finding all Maximal Homogeneous Clique Sets	44
4.2.1 Algorithm generate and test	44
4.2.2 Enumeration tree pruning techniques	48
4.2.3 Implementation	53
4.3 Experiments	54
4.3.1 Scientific collaboration network dataset	54
4.3.2 Interpretation of MHCSs from DBLP	55
4.3.3 Performance study on DBLP datasets.	57
4.3.4 Evaluation on synthetic datasets	60
4.3.5 Comparison of the prunings to baseline algorithms	61
5 MINING COLLECTIONS OF k-PCS	65

5.1	Pattern definition	65
5.2	Finding all CoHoP patterns	66
5.2.1	A naive algorithm	66
5.2.2	Enumeration tree pruning techniques	67
5.2.3	Algorithm description	68
5.2.4	Implementation	69
5.3	Experiments	69
5.3.1	Illustration of the patterns interest	70
5.3.2	Performance study	72
5.3.3	Evaluation on synthetic datasets	75
5.3.4	Comparison with baseline algorithm	75
	CONCLUSION	79
IV APPLICATION TO MOLECULAR BIOLOGY		81
	INTRODUCTION	83
6	APPLICATION TO MOLECULAR BIOLOGY	85
6.1	The dataset	85
6.2	Evaluation of the pattern collection	85
6.2.1	The L2L Measure	85
6.2.2	Global evaluation of complete collections of patterns	88
6.2.3	Interpretation	88
6.3	Performance evaluation	94
	CONCLUSION	99
V CONCLUSION		101
	Summary of our contributions	103
	Future directions of work	103
APPENDIX		107
A	DATASET DESCRIPTION	109
A.1	The BioData datasets	109
A.2	The DBLP datasets	109
B	A PATTERN EXTRACTION/VISUALISATION SOFTWARE	115
B.1	Presentation of the interface	115
B.2	Extraction of the patterns	116
B.3	Browsing the collection of patterns	116
B.4	Pattern visualisation	117
B.4.1	Centrality measures	117
B.4.2	Visualisation parameters	117
B.4.3	Layout	117
C	TABLE OF SYMBOLS	119
C.1	Notation in the binary relation setting	119
C.2	Notation in the graph setting	119
C.3	Notation in the Boolean attributed graph setting	120
VI BIBLIOGRAPHY		121

LIST OF FIGURES

Figure 1	An example of attributed graph.	4
Figure 2	The graph part corresponding to the attributed graph presented Figure 1. The solid lines structure is formed by vertices being pairwise connected.	5
Figure 3	An example collection of subgraphs illustrating the type of patterns we propose to extract.	7
Figure 4	An example of binary relation over the domain $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$ and codomain $\mathcal{I} = \{i_1, i_2, i_3, i_4, i_5\}$. A value 1 at the intersection of a row and a column depicts the fact that the corresponding object and item are related in the relation, otherwise they are not related. The order of the rows and columns is arbitrary.	13
Figure 5	An example of graph, with the set of vertices $\{A, B, C, D, E\}$	19
Figure 6	Example of path graph.	24
Figure 7	An example of attributed graph with vertices $\{A, B, C, D, E\}$ and three attributes: colour $\in \{\text{red, yellow, blue, purple}\}$, age $\in \mathbb{R}$, and music $\in \{\text{pop, jazz, classic, rock}\}$	27
Figure 8	An example of Boolean attributed graph with vertices $\{A, B, C, D, E\}$ and five Boolean attributes: pop, jazz, rock, pop, blues. Only attributes having value True are represented in the graph.	28
Figure 9	Example of Boolean attributed graph representing a social network (same as Figure 1).	38
Figure 10	Example of MHCS pattern in the Boolean attributed graph presented in Figure 9.	39
Figure 11	Example of Boolean attributed graph. Vertices are identified by capital letters and attributes are identified by attributes $a_i, i \in \{1, \dots, 6\}$	41
Figure 12	Enumeration tree corresponding to the attributed graph presented Figure 11.	47
Figure 13	Enumeration tree with pruning techniques 1, 2, 3, 4, and 5 corresponding to the attributed graph presented Figure 11 for constraints $C_2^{\text{hom}_0}$ and $C_{2,3}^{\text{clique}}$	50
Figure 14	A pattern related to conferences INFOCOM and SenSys. Each colour denotes a clique of at least κ vertices. A vertex in several colours is contained in multiple cliques. Vertices in light grey are not contained in a clique of at least κ vertices.	56
Figure 15	A pattern related to conferences ACM SIGMOD, CIKM, EDBT, ICDE, ICDM, and SDM. Each colour denotes a clique of at least κ vertices. A vertex in several colours is contained in multiple cliques. Vertices in light grey are not contained in a clique of at least κ vertices.	56

Figure 16	Runtime for different sets of parameters on DBLP ₁ , DBLP ₂ , and DBLP ₃	58
Figure 17	Number of MHCS for different sets of parameters on DBLP ₁ , DBLP ₂ , and DBLP ₃	59
Figure 18	Runtime for the collections of datasets $S_{\#vert}$, S_{avgDeg} , $S_{\#attr}$, S_{avgAtt} and $S_{\#hcs}$	62
Figure 19	Runtime for different pruning techniques on the datasets DBLP ₂ and DBLP ₃ . The scale is logarithmic for the runtime.	64
Figure 20	A Boolean attributed graph illustrating the CoHoP patterns. Vertices are identified by capital letters and attributes are identified by a_i , $i \in \{1, \dots, 5\}$. Only attributes having value True are displayed.	65
Figure 21	Two patterns extracted from DBLP ₃ with $k = 4$, $\gamma = 7$, and $\alpha = 3$. Each colour corresponds to a k -PC. A vertex in several colours is contained in multiple k -PCs.	71
Figure 22	Runtime for different sets of parameters on DBLP ₁ , DBLP ₂ , and DBLP ₃	73
Figure 23	Number of CoHoP for different sets of parameters on DBLP ₁ , DBLP ₂ , and DBLP ₃	74
Figure 24	Runtimes for the extraction of CoHoP patterns in the collections of datasets $S_{\#vert}$, S_{avgDeg} , $S_{\#attr}$, S_{avgAtt} and $S_{\#CoHoP}$	76
Figure 25	Runtime for the extraction of CoHoP patterns using different pruning techniques on DBLP ₂ and DBLP ₃ . The scale is logarithmic for the runtime.	78
Figure 26	Screen copy from SQUAT and STRING databases, showing information about gene CRX (HUGO nomenclature).	86
Figure 27	Cumulative distributions of p-values (on dataset BioData ₄₀₀ with α , γ and κ varying) in number of MHCSs extracted (left column) and in percentage of the MHCSs extracted (right column). P-value scale is logarithmic. The vertical dotted line corresponds to the significance level using Bonferroni correction (<i>i.e.</i> , $\approx 2.4 \times 10^{-5}$).	89
Figure 28	Cumulative distributions of p-values (on dataset BioData ₄₀₀ with α , γ and κ varying) , in number of CoHoPs extracted (left column) and in percentage of the CoHoPs extracted (right column). P-value scale is logarithmic. The vertical dotted line corresponds to the significance level using Bonferroni correction (<i>i.e.</i> , $\approx 2.4 \times 10^{-5}$).	90
Figure 29	MHCS extracted from dataset BioData ₄₀₀ with $\alpha = 3$, $\kappa = 4$ and $\gamma = 3$. All genes are overexpressed in 3 biological situations corresponding to normal activities of retinal cells.	92

Figure 30	MHCS extracted from dataset BioData ₄₀₀ with $\alpha = 3$, $\kappa = 4$ and $\gamma = 3$. All genes are overexpressed in 4 biological situations corresponding to normal activities of white blood cells. In the table, <i>signal transduction</i> and <i>regulation process</i> are short names for respectively <i>small GTPase mediated signal transduction</i> and <i>positive regulation of cytokine biosynthetic process</i>	93
Figure 31	A CoHoP extracted from dataset BioData ₄₀₀ with $\alpha = 4$, $k = 3$ and $\gamma = 3$. The genes are overexpressed in four biological situations related to breast carcinoma cell line, ZR75.	94
Figure 32	A CoHoP extracted from dataset BioData ₄₀₀ with $\alpha = 4$, $k = 3$ and $\gamma = 3$. The genes are overexpressed four biological situations related to antigen-purified CD14+ monocytes.	95
Figure 33	Extraction runtime for MHCSs and CoHoPs using different sets of parameters on BioData ₁₅₀ and BioData ₄₀₀	96
Figure 34	Number of extracted MHCSs and CoHoPs using different sets of parameters on BioData ₁₅₀ and BioData ₄₀₀ . The line is not drawn if no pattern are output for a set of parameters.	97
Figure 35	A general and a more specific CoHoP in a bibliographic dataset.	104
Figure 36	Example of subgraph where all vertices share two common attributes except vertex E.	105
Figure 37	Example of multidimensional network with attributes associated to the nodes.	106
Figure 38	The multidimensional network presented Figure 37 where edge labels are encoded by means of Boolean attributes in an attributed graph.	106
Figure 39	Example of structure which might be found in the graph presented Figure 38. It is formed by two groups of three vertices sharing two vertex attributes (rock and jazz) and two edge attributes (friend and colleague).	106
Figure 40	The largest connected component of BioData ₄₀₀ . Each colour corresponds to a module according to the modularity definition of [9].	110
Figure 41	Degree and attribute distribution for the datasets BioData ₁₅₀ and BioData ₄₀₀ . All the scales are logarithmic.	110
Figure 42	The largest connected component of DBLP ₃ . Each colour corresponds to a module according to the modularity definition of [9].	112
Figure 43	Degree and attribute distribution for the datasets DBLP ₁ , DBLP ₂ , and DBLP ₃ . All the scales are logarithmic.	113
Figure 44	The different parts of the interface	115
Figure 45	The collection of patterns view.	116
Figure 46	Panels used to customize pattern visualisation.	117
Figure 47	A CoHoP pattern displayed using the five available layouts.	118

LIST OF TABLES

Table 1	Measures describing datasets DBLP ₁ , DBLP ₂ , and DBLP ₃	55
Table 2	Memory consumption over all experiments reported Figure 16.	57
Table 3	Parameters used to generate the synthetic datasets.	61
Table 4	The five set of parameters with α between 3 and 9, γ between 5 and 9, and k between 3 and 6 that lead to a number of CoHoPs between 50 and 100 in DBLP ₃	70
Table 5	Memory consumption over all experiments reported in Figure 22.	72
Table 6	Parameters used to generate the synthetic datasets.	75
Table 7	Several characteristics of the datasets BioData ₁₅₀ and BioData ₄₀₀	87
Table 8	Main characteristics of the datasets BioData ₁₅₀ and BioData ₄₀₀ . Dashes denote prohibitive runtime or memory consumption.	109
Table 9	Main characteristics of the datasets DBLP ₁ , DBLP ₂ , and DBLP ₃ . Dashes denote prohibitive runtime or memory consumption.	111
Table 10	Summary of the notation used in the binary relation setting	119
Table 11	Summary of the notation used in the graph setting	119
Table 12	Summary of the notation used in the Boolean attributed graph setting	120

Part I

INTRODUCTION

CONTEXT

Scientific research is built from many previous results and the results presented in this thesis are not an exception. This work is based on previous results in the data mining community. This domain of study has emerged in the nineties at the boundaries of several other fields of research. Among these fields, probably the main ones were database, artificial intelligence and statistics. At that time, one of the main objective of data mining was to support the analysts in the knowledge discovery process.

Since then, while the original objectives of data mining remain, new challenges have occurred [36, 83]. The results presented in this thesis deal mainly with four challenges presented in the next paragraphs. However the data mining community has also to deal with other problems such as privacy, data visualisation, or data distribution over multiple computers, that are not tackled in this work.

Performance scaling

The efficiency challenge is strongly tied with one of the objective of data mining, being able to treat large and continuously growing volumes of data. The advance in hardware is of little help for this aim. Many algorithms in data mining have an exponential complexity with respect to the size of the data. For example, it is the complexity of most algorithms performing subset enumeration. Reducing algorithm complexity is then always a great challenge for the data mining community.

Domain knowledge

Another objective is to give the possibility to the experts to use their knowledge during the extraction process. Using this knowledge allows not only to get more meaningful results but it might also be used to avoid the cost of finding something which is not relevant for the expert. As a consequence, this challenge is linked with the efficiency. Being able to exploit expert knowledge in the extraction process is still an open problem. A common solution toward this aim is to specify the kind of information one wants to find by means of constraints that are then used to focus on more relevant results. Moreover, these constraints can also be used actively during the extraction to reduce the computational cost.

The inductive query framework proposed in [41] is an attempt to address this problem in a general manner.

Handling imperfect data

Another great challenge is to take into account missing values in the data. Several factors can generate missing values. This can happen, for example, during the data collection process. This can also be due to the data preprocessing. Indeed the raw data are usually transformed before they can be used by a data mining algorithm. Such preprocessing (e.g., binarization) can lead to loss of information and consequently to missing values. Taking into account the fact that values may be missing is then important.

Analysis of complex data

The last problem tackled by this thesis is the analysis of complex data objects, like web pages, genes or spatio temporal objects descriptions. Among the data model used, graphs have been shown to be a very general one, useful to encode many datasets. A large number of

data mining algorithms have already been proposed for the analysis of graphs. More recently, several works have been dedicated to the analysis of graph extensions as for example dynamic graphs [74], multi dimensional graphs [8] or attributed graphs.

In this thesis, our object of study is this last extension mentioned: the attributed graphs. For short, an attributed graph is a graph (*i.e.*, a collection of vertices connected by edges) such that properties are associated to each vertex. The properties can have different domain of values, for example Boolean, numeric (*e.g.*, age, weight), or categorical (*e.g.*, season, colour). Our work will consider the case where the domain of the attributes is Boolean.

In the literature, the term node is sometimes used instead of vertex. In this thesis, we will use the term vertex.

An example of attributed graph is given in Figure 1. These data represent a group of individuals with their relationships and their musical tastes. The kind of relationships can be for example friendship, geographic closeness, or being member of the same organization. In the attributed graph representation of these data, each person corresponds to a vertex denoted by a capital letter. The relationships between persons are represented by edges. The musical tastes (*e.g.*, rock, pop) of each person is encoded by means of attributes having either value True or False (*i.e.*, the domain of the attributes is Boolean). A person associated to an attribute having value True denotes the fact that this person enjoys the corresponding style of music. Note that in Figure 1, only attributes having value True are represented. For example, the person corresponding to vertex G (top right of the figure), is in relation with the persons denoted by vertices C, E, F, and H and enjoys rock, folk and jazz music styles.

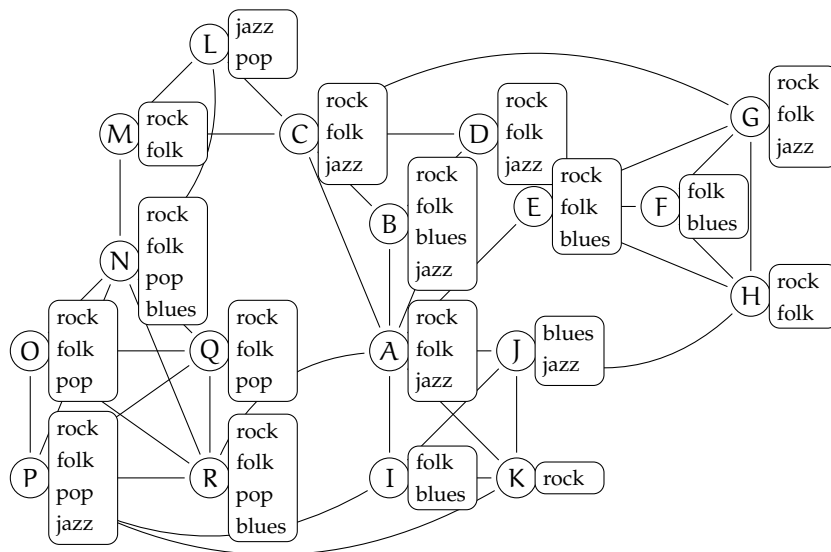


Figure 1.: An example of attributed graph.

To sum up, the context of this thesis is the analysis of attributed graphs with the objectives of being efficient, being able to handle missing values, and allowing the experts to use their knowledge during the extraction process. Given this context, the next section presents the problem tackled in this thesis.

PROBLEM

In this thesis, we focus on providing a method for the discovery of structures in attributed graphs. This problem specification is very general and must be defined more precisely. In fact, it raises the question “what can be considered as a relevant structure in an attributed graph?”

An example of well studied structure is a collection of properties or *items* shared by the same set of objects. In a dataset of persons (the objects) together with their musical tastes (the items), such a structure corresponds to a set of music styles enjoyed by a group of persons. This structure formed by a set of items, usually named *itemset*, occurring frequently with a set of objects, has been studied in data mining under the name *frequent itemsets*, and its extensions *frequent closed itemset* [70] or *error tolerant frequent closed itemsets* [18] among others.

Structures formed by groups of objects such that there are many connections between them have also been well studied in the context of graphs. Considering only the graph part of the attributed graph presented in Figure 1, Figure 2 highlights such collection of objects. The notion of *many connections* has been defined in several ways. One of these definitions is the notion of *cliques* [52] (a collection of objects all pairwise connected, as the subgraph in solid lines depicted in Figure 2). Since this definition is very restrictive, several extensions have been proposed to allow missing connections within the structure, for example *quasi-cliques* [51] or *k-clique percolated components* [67].

Here, the term *structure* is used in a general sense, and does not refer to the discovery of structures in documents as structure mining in XML documents.

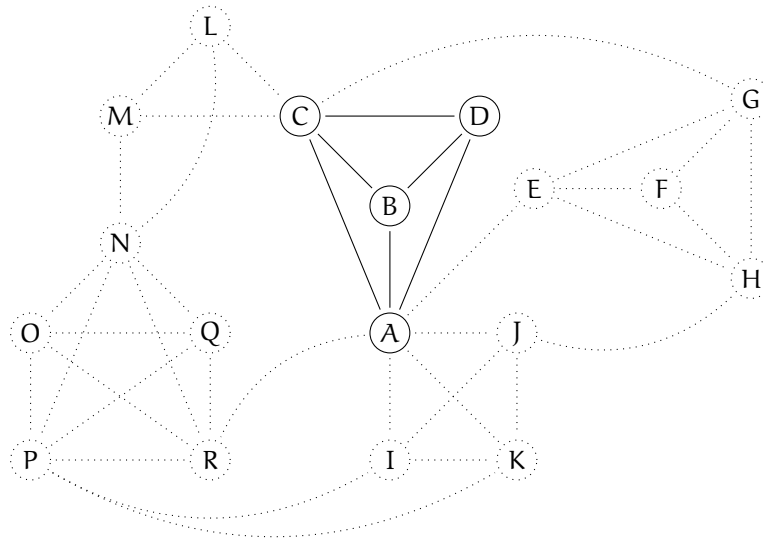


Figure 2.: The graph part corresponding to the attributed graph presented Figure 1. The solid lines structure is formed by vertices being pairwise connected.

In this thesis, we focus more particularly on attributed graphs. As said previously, an attributed graph can be viewed as entities which are both in relationship (*i.e.*, the graph) and associated to several properties (*i.e.*, the attributes). Intuitively, from what have been previously considered, an interesting structure could be a collection of objects sharing attribute values and being strongly connected. Such structure has been studied for example in [58] and can be considered as local in the sense that it is usually formed by a small number of objects

compared to the size of the whole dataset. However, it is possible to consider structures at an intermediate level of analysis, even if there is no strict definition of what is an intermediate level. In this thesis, we consider that a structure formed by several dense subgraphs can reasonably be placed at an intermediate level of analysis. Our work is placed at such intermediate level. More precisely, our structure of interest is formed by *several* groups of objects strongly connected and similar with respect to the values of their attributes.

CONTRIBUTION

Our main contribution consists in the definition of two classes of patterns used to discover collections of dense subgraphs sharing similar Boolean attributes. In Figure 3 the highlighted collection of three groups formed by densely connected vertices sharing attributes rock and folk illustrates the kind of patterns we propose to extract.

The first class of patterns, named Maximal Homogeneous Clique Set (MHCS), is based on cliques. To the best of our knowledge, it is the first attempt to find collections of dense subgraphs. The second class of patterns is named Collection of Homogeneous k-clique Percolated components (CoHoP). This class of patterns allows to take into account missing values in the data in the sense that the vertices in a subgraph are not required to be fully pairwise connected.

In order to compute the collections of MHCS and CoHoP patterns, we propose a sound and complete algorithm for each class of patterns. These algorithms take advantage of several pruning techniques based on constraint properties in order to reduce the search space. We also provide formal proof of correctness for these algorithms.

We present experimental results in two real world contexts, a scientific collaboration network and a protein-protein interaction network. These experiments demonstrate the practical interest of these classes of patterns. We also study the performances of our algorithms using large real and synthetic datasets. The results show that our approaches are able to handle real size datasets and scale well with respect to several attributed graph characteristics.

We also develop a pattern extraction and visualisation software. This tool allows to filter the collection of patterns and highlight vertices with respect to several graph characteristics.

The MHCS patterns have been introduced in [59, 61] and the CoHoP patterns in [62]. Their application to protein-protein interaction networks have been presented respectively in [60] and [63].

ORGANIZATION OF THE THESIS

In the following part (Part ii), we propose a state of the art on the analysis of attributed graphs using data mining techniques. Since the interest for such context is rather recent in data mining, only few approaches have been proposed. In order to get a broader overview, we also consider the discovery of groups of properties in binary relations (Section 1) and the discovery of groups of connected objects in graphs (Section 2). In the attributed graph context (Section 3), we present existing approaches to find groups either at a global level (clusters) or at a more local level (dense subgraphs).

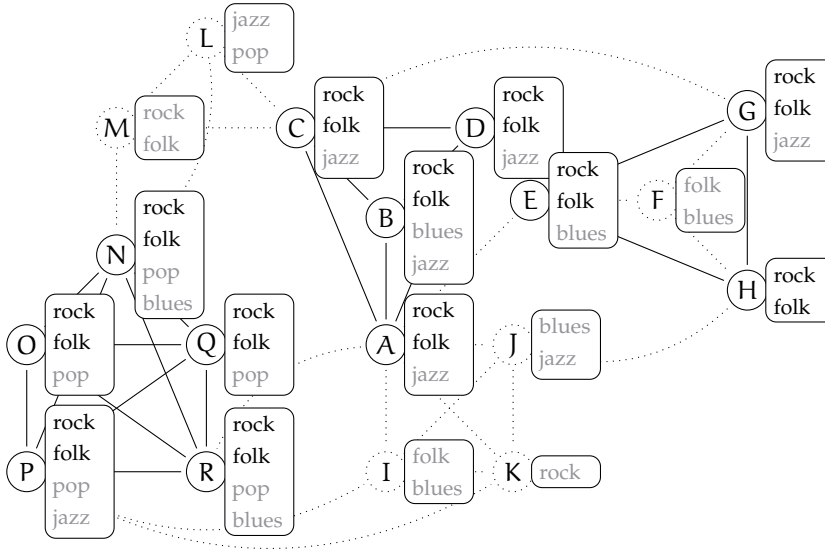


Figure 3.: An example collection of subgraphs illustrating the type of patterns we propose to extract.

Part [iii](#) details the main contributions of this thesis. Section [4](#) focuses on the Maximal Homogeneous Clique Sets and Section [5](#) on the Collection of Homogeneous k -clique Percolated components. The organization of these two sections are similar. First, we present the definition of our classes of patterns. Next, we present an extraction algorithm using several pruning techniques and demonstrate their safety. Finally, we present experimental results on bibliographical data in order to give practical examples of patterns in real datasets and study the scalability of our approaches on synthetic datasets.

Part [iv](#) proposes an application of both MHCS and CoHoP patterns in the context of molecular biology. Using a domain specific measure, we experimentally show that a large number of extracted patterns contain information in agreement with previous knowledge in biology. We also provide biological interpretations, obtained in collaboration with a domain expert, for several MHCS and CoHoP patterns. Finally, we present the performances of our algorithms on these datasets for various settings.

Part [v](#) concludes with a short summary of the results, and proposes several possible future works extending the results presented in this thesis.

Appendix [A](#) gives several characteristics describing the datasets used in the previous experiments. Appendix [B](#) describes the software which have been developed in order to extract and visualise collections of patterns. Appendix [C](#) details the symbols used in this thesis. Finally, the references are given in Part [vi](#).

Part II

STATE OF THE ART

OUTLINE

In this state of the art, we present previous works regarding the analysis of attributed graphs. Since the interest for this task is rather recent, we also consider other contexts to provide a broader overview. More precisely, we present techniques to find groups of properties, the so-called itemsets, and to find groups of objects connected in a graph.

In Section 1, we present classes of local patterns in the binary relation setting. These classes of patterns aim at finding set of attributes (the items) shared by several objects. Even though the data mining tasks have evolved since these first results, many recent approaches still share ideas with these works.

In Section 2, we consider the graph setting. In such context, we first present several graph measures used to characterize the structure of a graph. Then, we present three classes of patterns corresponding to groups of connected objects, namely cliques, quasi-cliques, and k -clique percolated components. The objective of this section is to give an overview of structures commonly used to exhibit dense subgraphs.

Section 3 presents several data mining tasks in the context of attributed graphs with a more general scope, from clustering to local pattern extraction. The objective of this section is to present the existing approaches to find groups in attributed graphs.

For each context, we discuss the efficiency of the presented approaches and the techniques used to handle missing values.

1.1 THE BINARY RELATION CONTEXT

A binary relation is used to depict the fact that an element from a set is related to an element from another set. More precisely, a binary relation consists in a collection of ordered pairs defined as follows.

Definition 1 (Binary Relation) *Given two arbitrary sets $\mathcal{O} = \{o_1, \dots, o_n\}$ and $\mathcal{J} = \{i_1, \dots, i_m\}$ called respectively domain and codomain, a binary relation \mathcal{R} is a subset of the Cartesian product $\mathcal{O} \times \mathcal{J}$, where the Cartesian product of \mathcal{O} and \mathcal{J} is*

$$\mathcal{O} \times \mathcal{J} = \{(o_1, i_1), \dots, (o_1, i_m), \dots, (o_n, i_1), \dots, (o_n, i_m)\}$$

Example 1 *Figure 4 depicts a binary relation over two sets \mathcal{O} and \mathcal{J} . The ordered pair (o_1, i_3) is in the relation whether (o_1, i_4) is not.*

In the data mining context, the elements of the domain \mathcal{O} and of the codomain \mathcal{J} are usually named respectively objects and items. A typical example of binary relation is a set of market baskets. In such data, the set of items is the products available and the set of objects is the customers. The relation will then associate a product i to a customer o when the customer o has bought product i . In such context, the binary relation presented Figure 4 depicts, for instance, the fact that customer o_4 bought products i_2 , i_3 , and i_5 .

	i_1	i_2	i_3	i_4	i_5
o_1	1	1	1		1
o_2		1	1	1	
o_3	1			1	
o_4		1	1		1

Figure 4: An example of binary relation over the domain $\mathcal{O} = \{o_1, o_2, o_3, o_4\}$ and codomain $\mathcal{J} = \{i_1, i_2, i_3, i_4, i_5\}$. A value 1 at the intersection of a row and a column depicts the fact that the corresponding object and item are related in the relation, otherwise they are not related. The order of the rows and columns is arbitrary.

In the following, we will consider a binary relation \mathcal{R} defined over two arbitrary sets \mathcal{O} and \mathcal{J} named respectively objects and items. Let us introduce some preliminary definitions. An *itemset* is a subset of \mathcal{J} and a *k-itemset* is an itemset of size k . The two following functions f and g are defined to associate a set of items (respectively objects) to a set of objects (respectively items).

Definition 2 (Functions f and g) *Let \mathcal{R} be a binary relation defined over two sets \mathcal{O} and \mathcal{J} . The functions $f: 2^{\mathcal{O}} \rightarrow 2^{\mathcal{J}}$ and $g: 2^{\mathcal{J}} \rightarrow 2^{\mathcal{O}}$ are defined as:*

$$f(\mathcal{O}) = \{i \in \mathcal{J} \mid \forall o \in \mathcal{O}, (o, i) \in \mathcal{R}\}$$

$$g(\mathcal{I}) = \{o \in \mathcal{O} \mid \forall i \in \mathcal{I}, (o, i) \in \mathcal{R}\}$$

Given I an itemset and O a set of objects, $f(O)$ is the set of items associated to all objects in O and $g(I)$ is the set of objects associated to all items in I . The *support* of an itemset I , denoted $\text{supp}(I)$, is the number of objects related to all items in I , *i.e.*, $\text{supp}(I) = |g(I)|$.

Example 2 In the binary relation presented Figure 4, $f(\{o_1, o_2\}) = \{i_2, i_3\}$, $g(\{i_2, i_3\}) = \{o_1, o_2, o_4\}$, and $\text{supp}(\{i_2, i_3\}) = 3$.

1.2 LOCAL PATTERNS IN BINARY RELATIONS

An exhaustive presentation of local patterns in binary relations is out of the scope of this state of the art. Instead, in the next subsections three well studied examples of local patterns are described, namely frequent itemsets, frequent closed itemsets and fault-tolerant closed itemsets. The reader interested in a more complete survey on local patterns mining in binary relations can refer to [33]. Within this presentation, we will focus on the algorithmic aspects of several local pattern mining algorithms, and compare the relative interest of the presented classes of patterns.

1.2.1 Frequent itemsets

A frequent itemset as introduced in [3] is a set of items co-occurring frequently in the data. This somehow trivial definition has produced one of the most studied local pattern in data mining. An itemset is considered *frequent* if it is formed by items shared by a set of objects sufficiently large.

Definition 3 (Frequent itemset) Let $\sigma \in \mathbb{N}$ be a minimal support threshold. An itemset $I \subseteq \mathcal{J}$ is a frequent itemset if and only if the frequency constraint C_σ^{freq} is satisfied, with $C_\sigma^{\text{freq}}(I) \equiv \text{supp}(I) \geq \sigma$.

Example 3 In the binary relation presented Figure 4, with a minimum support threshold $\sigma = 2$, the itemset $\{i_2, i_3\}$ is frequent while $\{i_1, i_2\}$ is not.

The first application to frequent itemsets, proposed together with the problem of mining frequent itemsets, was the discovery of association rules. An association rule can help to discover relationships between entities in a dataset. In the market basket context, an association rule represents the fact that, when one buys some given products, then it is likely that she/he will also buy some other products indicated by the rule. As it is possible to derive efficiently the collection of association rules from the collection of frequent itemsets, it has been a major application of frequent itemset mining. A survey on association rule mining is proposed in [40].

Mining frequent itemsets

Shortly after the publication of the introductory article on the frequent itemsets problem, Agrawal et al. proposed the Apriori algorithm to compute more efficiently the complete collection of frequent itemsets [2]. The general idea of Apriori is to generate a set of candidate itemsets, check whether or not they satisfy the frequency constraint, and build new candidate itemsets from the previous frequent itemsets. More formally, let C_k and F_k be respectively the collection of candidate k -itemsets and the collection of frequent k -itemsets. The Apriori algorithm performs the following four steps iteratively:

The minimum support threshold is sometimes given as a ratio relative to the number of objects in the relation.

1. Compute the support of each itemset in C_k
2. Add to F_k the itemsets in C_k verifying C_σ^{freq}
3. Generate C_{k+1} from F_k
4. Increment k and go to 1.

This process is described as Algorithm 1. The algorithm starts with $k = 1$ (C_k is the collection of all singletons itemsets) (lines 1 and 2). Once a value of k for which C_k is empty is found (line 3) the algorithm stops and the complete collection of frequent itemsets is the collection of frequent itemsets found for each values of k (line 7). As all candidate k -itemsets are enumerated before $(k + 1)$ -itemsets, the enumeration is performed level-wise. This algorithm uses the anti-monotonicity property of the support, *i.e.*, if an itemset is not frequent, then none of its superset are frequent. From this property, it is not necessary to construct candidate itemsets which are supersets of a non frequent itemset. Therefore, the $(k + 1)$ -itemset candidate generation (line 5) is performed by computing the union of frequent itemsets sharing the same $k - 1$ prefix with respect to an arbitrary order defined over the items. To compute the support of each itemset (line 4), Apriori scans the set of all objects, and increment the support of an itemset I for each objects o such that $I \subseteq f(\{o\})$.

The anti-monotonicity property of the support is also named downward closure in the literature.

Algorithm 1: Apriori

Input: $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{J}$, σ

- 1 $k \leftarrow 1$
- 2 $C_k \leftarrow \{ \{i\} \mid i \in \mathcal{J} \}$
- 3 **while** $C_k \neq \emptyset$ **do**
- 4 $F_k \leftarrow \{ C \in C_k \mid C_\sigma^{\text{freq}}(C) \}$
- 5 Generate C_{k+1} from F_k
- 6 $k \leftarrow k + 1$
- 7 **output** $\bigcup_{x=1}^{k-1} F_x$

Even though this algorithm has received a lot of attention from the data mining community, it suffers from several drawbacks. Consequently, there have been many attempts to devise more efficient algorithms for the frequent itemset mining task. For example, the algorithm Eclat [91] improves the computation of the support. Instead of scanning the collection of objects, it is generally more efficient to compute the support of an itemset using the intersection of the lists of objects associated to some of the subsets of the itemset. In this case, keeping in memory all objects associated to 1-itemsets and 2-itemsets could exceed main memory size. Fortunately, using a depth-first strategy can greatly reduce the required amount of memory needed, especially for small values of k .

Another great improvement of Apriori is the algorithm FP-growth proposed in [37]. Most of the time, it can reduce considerably the runtime by representing the dataset using a frequent pattern tree structure (FP-tree) and performing projections of the database over the enumerated itemsets.

1.2.2 Frequent closed itemsets

In many practical applications, the collection of frequent itemsets can be extremely large. One of the reason for this behaviour is the fact that every subset of a frequent itemset is also frequent. Therefore, the collection of frequent itemsets is usually too large for human browsing. Another consequence of this behaviour is the prohibitive extraction time for some datasets. The *frequent closed itemsets* defined as follows allow to diminish these drawbacks.

Definition 4 (Closed itemset and frequent closed itemset) *Let I be an itemset, $f \circ g(I)$ is called the closure of I . The set I is a closed itemset if and only if the following closure constraint C^{closed} is satisfied:*

$$C^{\text{closed}}(I) \equiv f \circ g(I) = I$$

The itemset I is a frequent closed itemset if and only if it satisfies the conjunction of constraints $C^{\text{closed}}(I) \wedge C_{\sigma}^{\text{freq}}(I)$, with $\sigma \in \mathbb{N}$ a minimal support threshold.

Example 4 *In the binary relation presented Figure 4, since $g(\{i_2, i_3\}) = \{o_1, o_2, o_4\}$ and $f(\{o_1, o_2, o_4\}) = \{i_2, i_3\}$, the itemset $\{i_2, i_3\}$ is closed.*

The closure operator ensures the maximality of the closed itemset among the other itemsets associated to the same set of objects, where maximal means not being the subset of any other set.

The introductory work on frequent closed itemsets in data mining has been proposed independently in [70] and [92]. However, in the field of formal concept analysis, the same structure was already studied under the name formal concept (see [28] for a survey on formal concept analysis).

An interesting property of the frequent closed itemsets is that they are an exact condensed representation of the frequent itemset. An exact condensed representation allows to represent exactly the same information about the support in a more succinct way. There have been several other works on condensed representations of frequent patterns, either exact or approximated, *e.g.*, maximal frequent itemset [15, 34, 90], non-derivable itemsets [16], key patterns [5], and δ -free-sets [13].

Mining frequent closed itemsets

The first frequent closed itemset mining algorithm has been proposed by Pasquier et al. in [70]. It constructs the collection of closed itemsets in an Apriori manner using frequent generators. A generator is an itemset not being a superset of any other itemset having the same closure. The frequent closed sets are the closures of these generators.

A very efficient algorithm is the CLOSET algorithm [71] that uses a depth-first strategy and extends the FP-Growth projection technique to mine frequent closed itemsets.

1.2.3 Frequent closed error-tolerant itemsets

While the frequent closed itemsets usually allow to reduce the collection of patterns by several orders of magnitude, most of the time the collection remains still too numerous to be exploited directly by a human. Indeed, in a collection of frequent closed itemset, many patterns are very similar to each other and are due to a few missing values.

For example, in the binary relation presented Figure 4, $\{i_2, i_3, i_5\}$ and $\{i_2, i_3\}$, associated respectively to the objects $\{o_1, o_4\}$ and $\{o_1, o_2, o_4\}$, are both closed itemsets. This is due to the absence of the ordered pair (o_2, i_5) in the relation. If (o_2, i_5) would have been present, then $\{i_2, i_3\}$ would not be closed while $\{i_2, i_3, i_5\}$ would still be a closed itemset.

In [82], the authors demonstrate that the number of frequent itemsets exponentially grows with the number of missing values. To take into account missing values in data and avoid pattern flooding, *closed error-tolerant itemsets* (closed ET-itemsets for short) have been proposed. The general idea is to allow a certain number of missing ordered pairs (o, i) among the items and the objects used to build a pattern. A closed ET-itemset is defined not as a single set of items but as an ordered pair formed by an itemset and a set of objects. Considering a closed ET-itemset $(O, I) \in 2^O \times 2^I$, the error tolerance threshold is a bounded number of missing ordered pairs in the relation $\mathcal{R} \cap (O \times I)$.

Several ways of counting the number of missing ordered pairs have been proposed. The error tolerance threshold can be either absolute or relative to the size of the pattern. The threshold can also be applied either to each object or item separately, or to the whole pattern. In the following definition the threshold is absolute and applied to each object and item separately.

Definition 5 (closed ET-itemset, frequent closed ET-itemset) *Given an error tolerance threshold $\delta \in \mathbb{N}$, an ordered pair $(O, I) \in 2^O \times 2^I$ is a closed ET-itemset if and only if it satisfies the two following constraints:*

$$\begin{aligned} C^{\text{connected}}(O, I) &\equiv \begin{cases} \forall o \in O, & |I \setminus f(\{o\})| \leq \delta \\ \forall i \in I, & |O \setminus g(\{i\})| \leq \delta \end{cases} \\ C^{\text{ET-max}}(O, I) &\equiv \forall (O', I') \in 2^O \times 2^I, (O, I) \neq (O', I'), \\ &\quad (O \subseteq O' \wedge I \subseteq I') \Rightarrow \neg C^{\text{connected}}(O', I') \end{aligned}$$

$(O, I) \in 2^O \times 2^I$ is a frequent closed ET-itemset if and only if it is a closed ET-itemset and $|O| \geq \sigma$, with $\sigma \in \mathbb{N}$ a minimal support threshold.

The $C^{\text{connected}}$ constraint ensures that for all items (respectively objects) in the pattern, the number of objects (respectively items) in the pattern which are not associated to it is at most δ . Note that while it is not the case here, the error tolerance threshold can be different for the set of items and the set of objects. The $C^{\text{ET-max}}$ constraint ensures the maximality of the pattern in the sense that there is no ordered pair formed by supersets satisfying the $C^{\text{connected}}$ constraint.

Example 5 *In the relation presented Figure 4, with an error tolerance threshold $\delta = 1$ and a minimum frequency threshold $\sigma = 2$, the ordered pair $(\{o_1, o_2, o_4\}, \{i_2, i_3, i_5\})$ is a frequent closed ET-itemset. Indeed, only the ordered pair (o_2, i_5) is missing in the relation to have all associations between $\{o_1, o_2, o_4\}$ and $\{i_2, i_3, i_5\}$. The ordered pairs $(\{i_2, i_3, i_5\}, \{o_1, o_4\})$ and $(\{i_2, i_3\}, \{o_1, o_2, o_4\})$ are not frequent closed ET-itemset since they are not maximal.*

1.2.4 Constraints on patterns and pattern sets

Constraints are important in local pattern mining tasks. On one hand, they allow the analyst to inject domain knowledge in the extraction

Missing values can have several sources, for example, it might be intrinsic to the system under study or due to the preprocessing, as for instance a binarization threshold used to obtain a binary relation from numerical data.

Using a threshold for the whole pattern has been shown to provide poor results in [76].

process. On the other hand, some constraint properties can be used to enumerate more efficiently the patterns. Regarding domain knowledge, the frequency constraint is very simple, and most of the time not sufficient. Other more complex constraints have been proposed in the literature (e.g. [81]). An example of such constraint is the area constraint which bounds the product of the number of items in an itemset and the number of associated objects. In most cases, such constraints do not satisfy to the anti-monotonicity property and require more sophisticated handling to use them actively in order to reduce the search. Difference classes of constraints have been studied along with techniques to use them in the enumeration process.

Most of these classes have been developed for itemset mining, and several of them have been reused and/or adapted to other kinds of local patterns (e.g., sequential patterns). Among the most representative earliest classes that have been studied, we have the monotone constraints and the convertible constraints. A monotone constraint is simply the negation of an anti-monotone constraint. For such a constraint, if it is satisfied by an itemset, this implies that all supersets of this itemset also satisfy the constraints. Handling these monotone constraints can lead to important execution time gains, in particular when using data reduction techniques [11, 12]. In the class of the convertible constraints, we find constraints that exhibit some monotonic or anti-monotonic behaviour with respect to the order of enumeration of the patterns. From an operational point of view, this means that there exists an enumeration ordering to produce the patterns to be tested that guaranties one of the following properties: for all patterns P' produced by expanding a pattern P , P' will be such that (1) if P satisfies the constraint then P' also satisfies it (monotonic behaviour), or (2) if P does not satisfy the constraint then the same holds for P' (anti-monotonic behaviour).

Since these works, other classes of constraints have been identified, with the aim to push (i.e., to use actively) more and more complex constraints, as for instance, the loose anti-monotone constraints [10], the flexible constraints [81], or the piecewise anti-monotone constraints [17].

The constraints are also used when considering a collection of patterns. This is one of the objectives of the approaches named pattern set [73] or pattern teams [46]. Such approaches aim at finding a collection of patterns which is relevant with respect to a given optimality criterion that can be expressed by means of constraints. For example, the coverage [30] or the minimum description length [78] criterion aim at finding a collection of patterns which best describes the data.

2.1 THE GRAPH CONTEXT

A graph consists in a set of vertices and a set of edges linking the vertices. It is generally used to depict the relationships between entities of the same type. Graphs have been widely used to model an important number of data. Typical applications using graphs are (non exhaustive list) the analysis of communication networks, disease spreading, social networks, protein protein interaction networks, World Wide Web, structure of chemical compounds. This wide range of applications has motivated a strong interest in the analysis of graphs. One of the oldest problem regarding graphs is the seven bridges problem of Königsberg published in 1741 [25]. Since then, many research fields including physics, biology, sociology, and computer science have been using graph theory to solve problems.

Various classes of graphs have been defined, for example directed or undirected, weighted or unweighted. In this section, we focus on undirected and unweighted graphs, however most of the work presented here have been extended to other classes of graphs. The next section will focus on attributed graphs, a class of graphs where information are associated to vertices. An undirected with no self-loop and unweighted graph is called simple graph [32]. Here we use the term graph for simple graph.

Definition 6 (Graph) A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an ordered pair formed by a set of vertices \mathcal{V} and a set of edges \mathcal{E} representing relationships between the vertices, where an edge is a set of two different vertices in \mathcal{V} .

Example 6 Figure 5 depicts a graph with the set of vertices $\{A, B, C, D, E\}$. The set $\{A, C\}$ is an edge of the graph while $\{A, E\}$ is not.

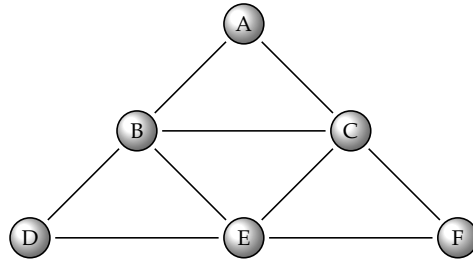


Figure 5.: An example of graph, with the set of vertices $\{A, B, C, D, E\}$.

We will use the following standard vocabulary in the context of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The *neighbourhood* of a vertex v , denoted $\Gamma(v)$, is the set of vertices directly connected to v , i.e., $\Gamma(v) = \{v' \in \mathcal{V} \mid \{v, v'\} \in \mathcal{E}\}$. The *degree* of a vertex v is $|\Gamma(v)|$. The *order* of a graph is its number of vertices, i.e., $|\mathcal{V}|$, while its *size* is $|\mathcal{E}|$. The *density* of a graph denoted by $\rho(\mathcal{G})$ is the ratio between the number of edges and the number of possible edges, i.e., $\rho(\mathcal{G}) = \frac{|\mathcal{E}|}{|\mathcal{V}| \cdot (|\mathcal{V}|-1)/2}$. The *subgraph induced* by a set of vertices $V \subseteq \mathcal{V}$, is denoted $\mathcal{G}[V]$ and is defined as \mathcal{G} restricted to the vertices in V and including only edges between vertices in V .

2.1.1 Graph measures

Before data mining techniques started to be adapted to study graphs, other fields of research were interested in the analysis of graphs. Several approaches focus on macroscopic properties which are of great use to understand the global structure of a graph.

A common approach in graph analysis is to study the *degree distribution*. It is the probability distribution of the vertex degree over the whole graph, sometimes presented as a cumulative degree distribution. It has been shown that for a large class of graphs the degree distribution follows a power law. Typical example of graphs having such degree distribution are the world wide web, metabolic networks, and telephone call graphs [66]. The *diameter* of a graph is the size of the longest shortest path between every pair of vertices. In other words, it is the minimum number of edges that must be traversed between the two most distant vertices.

Other approaches study how vertices tend to group together. The *clustering coefficient* is a common measure to quantify whether or not vertices tend to cluster together. It is defined as the ratio between the number of triangle (a triangle is three vertices pairwise connected) and the number of possible triangle containing a given vertex. For a vertex v , the clustering coefficient is

$$\frac{|{\{n_1, n_2\} \in \mathcal{E} \mid n_1 \in \Gamma(v) \wedge n_2 \in \Gamma(v)\}}|}{|\Gamma(v)| \cdot (|\Gamma(v)| - 1)/2}$$

The average clustering coefficient gives an overview for the whole graph. It is computed as the mean of the vertices clustering coefficient.

Other measures try to quantify the relative importance of a given vertex. They are called centrality measures. The *betweenness centrality* has been proposed in [26]. It is used to measure the overall importance of a vertex in a communication process between any two vertices in the graph. It is defined as the average number of shortest paths that pass through the vertex under study. Such measure is important for example in a computer network to capture the notion of bottleneck. The *closeness centrality* proposed in [75] quantify the time it will take for a vertex to communicate with all other vertices in the graph. For a vertex v it is defined as $\frac{|\mathcal{V}|-1}{\sum_{u \in \mathcal{V}, u \neq v} d(u, v)}$ where $d(u, v)$ denotes the length of the shortest path between vertices u and v .

2.2 LOCAL PATTERNS IN GRAPHS

While graph measures tend to reflect general topological characteristics of a given graph or vertex, it does not allow to automatically discover interesting subgraphs. Similarly to what has been presented in the binary relation context, local pattern mining in graphs allows to uncover potentially interesting structures in the data. Here we present three classes of local patterns in graphs, namely cliques, quasi-cliques and k -clique percolated components. We also present typical extraction algorithms for maximal cliques and k -clique percolated components.

2.2.1 Cliques and maximal cliques

A clique is usually defined as a set of vertices such that all vertices are pairwise connected in the graph. Such structure has been studied since 1935 [23] and many theoretical results have been obtained since then. From [56], the maximum number of maximal cliques in a graph with degree $n \geq 2$ is

$$\begin{cases} 3^{n/3} & \text{if } n \text{ modulo } 3 \text{ is equal to } 0 \\ 4 \times 3^{\lfloor n/3 \rfloor - 1} & \text{if } n \text{ modulo } 3 \text{ is equal to } 1 \\ 2 \times 3^{\lfloor n/3 \rfloor} & \text{if } n \text{ modulo } 3 \text{ is equal to } 2 \end{cases}$$

The formal definition of cliques and maximal cliques follows.

Definition 7 (Cliques and Maximal Cliques) *Given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a graph, $C \subseteq \mathcal{V}$ is a clique if and only if $\mathcal{G}[C]$ is a complete subgraph, i.e., all vertices are pairwise connected in \mathcal{G} . It is a maximal clique of \mathcal{G} if and only if it is not a subset of any other clique in \mathcal{G} .*

In the literature, a clique is also defined as a complete subgraph, i.e., including the edges. The term clique is also found in the sense maximal clique.

Example 7 *In the graph presented Figure 5, AB is a clique, but it is not maximal since it is a subset of the maximal clique ABC.*

The cliques capture strong structure in the graph since every vertex has to be connected with the others. For example, in a social network a clique would be a group of person where everyone has a relationship with all other members of the group. We will use the following vocabulary. A k -clique is clique with exactly k vertices, and a k -maximal clique is a maximal clique with at least k vertices. The collection of cliques, k -cliques, maximal cliques, and k -maximal cliques in a graph \mathcal{G} are denoted respectively $\mathcal{C}(\mathcal{G})$, $\mathcal{C}_k(\mathcal{G})$, $\mathcal{C}_{\max}(\mathcal{G})$, and $\mathcal{C}_{k\max}(\mathcal{G})$.

2.2.1.1 Mining maximal cliques

Several algorithms have been proposed to extract maximal cliques in a graph using different strategies [31, 53, 84]. Here, we will present the algorithm CLIQUES [84] which has a worst case time complexity of $O(3^{n/3})$ for a graph having degree n . This is optimal in the sense that in a graph containing n vertices, there can be no more than $3^{n/3}$ maximal cliques [56]. The standard technique to compute a maximal clique consists in expanding a set of vertices C known to be a clique (possibly a single node clique) with a vertex $v \in \mathcal{V}$ connected to all vertices in C , and repeat the same expansion on set $C \cup \{v\}$ until no new vertex can expand the current set. Let V_{ext} be the set of vertices that can be used to expand the current set C , i.e., $V_{\text{ext}} = \{v \in \mathcal{V} \setminus C \mid \forall c \in C, \{v, c\} \in \mathcal{E}\}$. Clearly, when V_{ext} is empty, then C is a maximal clique, otherwise, C can be extended with a vertex from V_{ext} to build a larger clique.

The CLIQUES algorithm is based on this idea, and follows an enumeration tree that expands each single vertex clique at the first level, in a depth-first way, and that stops an enumeration branch when a maximal clique is obtained. The CLIQUES algorithm extends the standard technique with two prunings as presented in Algorithm 2.

The first pruning consists in avoiding to add a vertex that has already been used to extend C in a previous branch of the enumeration tree. In order to do so, a set $\mathcal{V}_{\text{cand}}$ is used, containing the candidate vertices that are the vertices from V_{ext} which have not been previously used to extend C (line 8).

We can extend the algorithm to extract k -maximal cliques by testing line 2 that C satisfies $|C| \geq k$. Moreover, an additional pruning can be made, by requiring line 4 that the condition $|C \cup \mathcal{V}_{\text{cand}}| \geq k$ is also satisfied.

The second pruning technique reduces the number of vertices in V_{cand} . It is based on the following observation. Given C the clique considered at the current node of the enumeration tree and $u \in V_{\text{ext}}$ a vertex which can extend C , since all maximal cliques containing $C \cup \{u\}$ have been enumerated in the previous branches or will be enumerated in the subtree rooted at the current node, and because these maximal cliques contains also the vertices in V_{cand} being in the neighbourhood of u (i.e., vertices in $V_{\text{cand}} \cap \Gamma(u)$), then it is not necessary to extend C at the current node of the tree with a vertex from $V_{\text{cand}} \cap \Gamma(u)$. The corresponding pruning technique consists in selecting the vertex $u \in V_{\text{cand}}$ maximising the size of the intersection between $\Gamma(u)$ and V_{cand} (line 3) and to consider only the vertices from V_{cand} not being in the neighbourhood of u (lines 4 and 5) as candidates to extend C . Then, the enumeration goes on in the loop by extending C with each vertex in $V_{\text{cand}} \setminus \Gamma(u)$ (this includes u) on lines 6 and 7. Notice that choosing the vertex u that maximizes $|V_{\text{cand}} \cap \Gamma(u)|$ leads to the smallest size for $V_{\text{cand}} \setminus \Gamma(u)$, and thus to the smallest number of branches to expand the current node.

For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the parameters of the initial call to CLIQUES are $V_{\text{ext}} = \mathcal{V}$, $V_{\text{cand}} = \mathcal{V}$, and $C = \emptyset$.

Algorithm 2: CLIQUES

Input: $V_{\text{ext}}, V_{\text{cand}}, C$

```

1 if  $V_{\text{ext}} = \emptyset$  then
2   |  $C$  is a maximal clique
   else
3   |  $u = \arg \max_{v \in V_{\text{ext}}} |V_{\text{cand}} \cap \Gamma(v)|$ 
4   | while  $V_{\text{cand}} \setminus \Gamma(u) \neq \emptyset$  do
5   |   | Pick a vertex  $v$  from  $V_{\text{cand}} \setminus \Gamma(u)$ 
6   |   |  $C = C \cup \{v\}$ 
7   |   | CLIQUES( $V_{\text{ext}} \cap \Gamma(v), V_{\text{cand}} \cap \Gamma(v), C$ )
8   |   |  $V_{\text{cand}} = V_{\text{cand}} \setminus \{v\}$ 
9   |   |  $C = C \setminus \{v\}$ 

```

2.2.2 Quasi-cliques

Since in many cases there is missing edges in the data, or one might want to find dense but not necessarily complete subgraphs, it is necessary to define classes of patterns allowing a bounded number of missing edges. Two definitions for such class of patterns are commonly used: pseudo-cliques and quasi-cliques. The *Pseudo-clique* definition is based on a minimal density threshold applied on a subgraph as a whole, whereas the *quasi-clique* definition allows a bounded number of missing edges for each vertex. We recall the formal definitions of pseudo-cliques, quasi-cliques, and corresponding maximality.

Definition 8 (δ -pseudo-clique, maximal δ -pseudo-clique) *Given a minimal density threshold $\delta \in [0, 1]$, $C \subseteq \mathcal{V}$ is a δ -pseudo-clique if and only if the conjunction of constraints $C_{\delta}^{\text{p-clique}}(C) \wedge C^{\text{connected}}(C)$ is satisfied, where*

$$\begin{aligned}
 C_{\delta}^{\text{p-clique}}(C) &\equiv \rho(\mathcal{G}[C]) \geq \delta \text{ (with } \rho \text{ the density introduced p.19)} \\
 C^{\text{connected}}(C) &\equiv \forall v \in C, \Gamma(v) \cap C \neq \emptyset
 \end{aligned}$$

A δ -pseudo-clique is maximal if it is not a subset of another δ -pseudo-clique.

The $C_\delta^{\text{p-clique}}$ constraint requires that the density within the subgraph induced by the pseudo-clique is greater than the minimal density threshold δ . Since a subgraph can be dense but not being connected (for example, a graph formed by a large clique and another disconnected vertex), it is necessary to add the $C^{\text{connected}}$ constraint to ensure the connectivity of the subgraph induced by the pseudo-clique.

Example 8 In the graph presented Figure 5, with a minimal density threshold $\delta = 0.8$, ABCE is a δ -pseudo-clique since the induced subgraph has a density of $5/6$. Moreover it is maximal since adding either vertex D or F reduce the density under the density threshold.

Definition 9 (δ -quasi-clique, maximal δ -quasi-clique) Given $\delta \in [0, 1]$ a minimum degree threshold, $C \subseteq \mathcal{V}$ is a quasi-clique if and only if the conjunction of constraints $C_\delta^{\text{q-clique}}(C) \wedge C^{\text{connected}}(C)$ is satisfied, where

$$\begin{aligned} C_\delta^{\text{q-clique}}(C) &\equiv \min_{v \in C} (|\Gamma(v) \cap C|) \geq \lceil \delta \cdot (|C| - 1) \rceil \\ C^{\text{connected}}(C) &\equiv \forall v \in C, \Gamma(v) \cap C \neq \emptyset \end{aligned}$$

While parameter δ does not express a density threshold, one can derive a bound since the maximal number of missing edges in a δ -quasi-clique C is $\delta \cdot |C|$.

A δ -quasi-clique is maximal if it is not a subset of another δ -quasi-clique.

Example 9 In the graph from Figure 5, with a minimum degree threshold $\delta = 0.6$, BCDE is a δ -quasi-clique since all vertices are connected to at least two other vertices. It is also maximal since adding either A or F would violate $C_\delta^{\text{q-clique}}$.

The $C_\delta^{\text{q-clique}}$ constraint ensures that each vertex in a quasi-clique is connected to at least a given fraction of the vertices in the quasi-clique. For this reason, as argued in [80], it is usually preferable to extract quasi-cliques instead of pseudo-cliques.

2.2.2.1 Mining quasi-clique

Several algorithms have been proposed to mine pseudo-cliques [1] or quasi-cliques [51, 93]. Probably the most efficient quasi-clique mining algorithm is Quick presented in [51]. The enumeration technique is similar to the one presented in Algorithm 2, with additional pruning techniques allowing to reduce the number of vertices which can possibly extend a quasi-clique.

2.2.3 k-clique percolated components

The k-clique percolated components (named also k-clique percolation cluster [29] or k-clique-community [47]) have been defined in [67] to find communities in graphs. This class of patterns aims to merge cliques to form larger connected patterns. More precisely, the idea behind k-clique percolated components is to merge k-cliques sharing $k - 1$ vertices into a single pattern. It ensures that two patterns will not share more than a user defined number of vertices.

The definition of k-clique percolated component given in [20] can be reformulated as follows using an equivalence relation over the cliques.

Definition 10 (Adjacency relation) Let \mathcal{G} be a graph and \mathcal{R}_a be the adjacency relation over the k -cliques in \mathcal{G} . Two k -cliques are related by \mathcal{R}_a if and only if they have an intersection of at least $k - 1$ vertices. Let \mathcal{R}_a^t be the transitive closure of \mathcal{R}_a .

The relation \mathcal{R}_a is symmetric and reflexive, so \mathcal{R}_a^t is symmetric, reflexive, and transitive. Consequently, \mathcal{R}_a^t is an equivalence relation.

Definition 11 (k-clique percolated component (k-PC)) A k -clique percolated component (k -PC) is the union of all k -cliques in a class of equivalence over \mathcal{R}_a^t .

Example 10 Consider the 3-PC (i.e., $k = 3$) in the graph presented in Figure 5. The graph contains four 3-cliques, ABC, BCE, BDE, and CEF. Since ABC, BCE share two vertices ($|ABC \cap BCE| = k - 1$), they can be merged. However, this union would not be maximal since CEF share two vertices with BCE. Indeed, BCE also share two vertices with BDE and ABC, and thus the graph contains only one 3-PC, ABCDEF.

We will denote $\mathcal{C}_{kpc}(\mathcal{G})$ the collection of all k -PCs in an attributed graph \mathcal{G} . Note that this definition does not ensure that the induced subgraph is very dense. Consider the graph given Figure 6. The whole graph is a 2-PC, each 2-clique shares one vertex with an adjacent 2-clique. However, it is clear that such graph is not very dense. Instead of enforcing graph density, the k -PC definition enforces the fact that each vertex can be reached from any other through well connected subset of vertices [67].

Note that with $k = 2$, the 2-PCs are the connected components having at least 2 vertices.



Figure 6.: Example of path graph.

2.2.3.1 Mining k -clique percolated components

Together with the introduction of k -clique percolated components in [67], the authors proposed an algorithm to compute the k -PCs. Their algorithm computes the k -PCs using the following three steps:

1. Compute the collection of k -maximal cliques (i.e., the maximal cliques containing at least k vertices, as introduced p.21);
2. Build a binary matrix representing the adjacency relation between the k -maximal cliques;
3. Compute the connected components of the adjacency relation using the matrix.

The k -PCs are then these connected components. Note that since all k -cliques in a k -maximal clique are necessarily adjacent, these steps use directly the k -maximal cliques instead of the k -cliques. For simplicity, Algorithm 3 uses a binary relation instead of a matrix, however, since the order of the rows and columns in the matrix are arbitrary, the principle remain identical.

On line 1, the collection of k -maximal cliques is computed and stored in \mathcal{C}_{kmax} . The lines 2 to 5 build the adjacency relation \mathcal{R}_a by adding an ordered pair (C_1, C_2) in \mathcal{R}_a if C_1 and C_2 share at least $k - 1$ vertices. Lines 6 to 14 compute the k -PCs from \mathcal{R}_a by computing the transitive closure of the relation. The idea is to pick a first k -maximal clique

Any k -maximal clique mining algorithm, like CLIQUES presented previously, can be used to compute \mathcal{C}_{kmax} on line 1.

and expend it until there is no more adjacent k -maximal cliques (loop lines 10 to 13). A k -PC is then the union of all adjacent k -maximal cliques.

Algorithm 3: CFinder

Input: \mathcal{G}

```

1  $C_{kmax} \leftarrow \mathcal{C}_{kmax}(\mathcal{G})$ 
2 Let  $\mathcal{R}_a = \emptyset$  be a binary relation over  $C_{kmax} \times C_{kmax}$ 
3 for  $(C_1, C_2) \in C_{kmax} \times C_{kmax}$  do
4   if  $|C_1 \cap C_2| \geq k-1$  and  $C_1 \neq C_2$  then
5      $\mathcal{R}_a \leftarrow \mathcal{R}_a \cup \{(C_1, C_2)\}$ 
6 while  $C_{kmax} \neq \emptyset$  do
7   choose a  $k$ -maximal clique  $C$  in  $C_{kmax}$ 
8    $kcliques\_in\_kpc \leftarrow \emptyset$ 
9    $adj \leftarrow \{C\}$ 
10  while  $adj \neq \emptyset$  do
11     $kcliques\_in\_kpc \leftarrow kcliques\_in\_kpc \cup adj$ 
12     $C_{kmax} \leftarrow C_{kmax} \setminus adj$ 
13     $adj = \{C_2 \in C_{kmax} \mid \exists C_1 \in adj \text{ s.t. } (C_1, C_2) \in \mathcal{R}_a\}$ 
14  output  $\cup_{C \in kcliques\_in\_kpc}$ 

```

A second algorithm has been proposed in [47] to extract k -PCs. Instead of computing the k -maximal cliques, it computes the k -cliques using the following property. If there is an edge between two vertices u and v , and if there is a $(k-2)$ -clique C formed by the vertices being in the neighbourhood of both u and v , then $C \cup \{u, v\}$ is a k -clique. To use this property efficiently, the authors propose an incremental method. More precisely, the algorithm starts with a graph with no edge and it adds iteratively the edges from the original graph. For each edge $\{u, v\}$ it checks if a $(k-2)$ -clique is found in the neighbourhood shared by u and v . If so, this $(k-2)$ -clique can be extended by vertices u and v to form a new k -clique. Then, to extract the k -PCs from the k -cliques, the algorithm builds a graph where the vertices are $(k-1)$ -cliques and there is an edge between two vertices if the corresponding $(k-1)$ -cliques are part of the same k -clique. The k -PCs are then the connected components of this graph and an algorithm similar to Algorithm 3 lines 2 to 14 can be used to compute them. Depending on the graph to process, this algorithm can be faster than Algorithm 3. On the one hand, it replaces the computation of all maximal cliques (Algorithm 3 line 1) by an enumeration of the edges. On the other hand, the adjacency graph used can be larger and then can require more time to compute the connected components.

3.1 THE ATTRIBUTED GRAPH CONTEXT

Graphs where information are associated to vertices have been used in different works under various names including, for example, attributed graphs [80], itemset-associated graphs [77], graphs with feature vectors [58], informative graphs [57]. Even though the names are different, all these data types can be model using the attributed graphs.

For example, an itemset-associated graphs as defined in [77] is a simple graph where a set of labels is associated to each vertex. By encoding each label using a Boolean attribute it is possible to build an attributed graph where each vertex is attached to a set of attribute values, such that an attribute will have value True for a vertex if the corresponding label is associated to the vertex in the itemset-associated graphs, and False otherwise.

For the sake of simplicity, we will use the term *attributed graph* for such structure and define the corresponding attribute domain depending on the task performed.

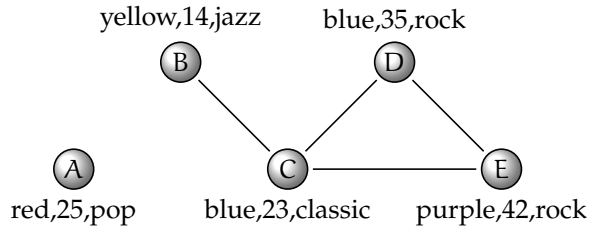


Figure 7.: An example of attributed graph with vertices $\{A, B, C, D, E\}$ and three attributes: colour $\in \{\text{red, yellow, blue, purple}\}$, age $\in \mathbb{R}$, and music $\in \{\text{pop, jazz, classic, rock}\}$.

Definition 12 (Attributed graph) An attributed graph is denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{F})$ where \mathcal{V} is the set of vertices, \mathcal{E} is the set of edges, $\mathcal{A} = \{A_1, \dots, A_n\}$ is the set of attributes, and $\mathcal{F} : \mathcal{V} \mapsto \text{dom}(A_1) \times \dots \times \text{dom}(A_n)$ is a function associating to each vertex a tuple of attribute values, with $\text{dom}(A_i)$ the domain of attribute A_i .

Since for many applications, the domain of the attributes can be restricted to Boolean values, we also define the *Boolean attributed graph* context. It is a specialisation of the attributed graphs where the domain of the attributes is $\{\text{True, False}\}$.

Definition 13 (Boolean Attributed graph) A Boolean attributed graph is denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \text{ATB})$ where \mathcal{V} is the set of vertices, \mathcal{E} is the set of edges, \mathcal{A} is the set of Boolean attributes, and $\text{ATB} : \mathcal{V} \mapsto 2^{\mathcal{A}}$ is a function associating to each vertex the Boolean attributes having value True for this vertex.

The vocabulary defined previously in the context of a simple graph remains identical in the context of an attributed graph. The following notation will be used in the context of a Boolean attributed graph. The

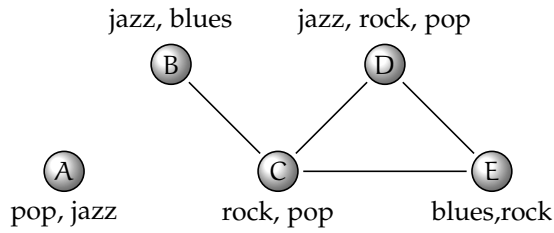


Figure 8.: An example of Boolean attributed graph with vertices $\{A, B, C, D, E\}$ and five Boolean attributes: pop, jazz, rock, pop, blues. Only attributes having value True are represented in the graph.

set of attributes having value True for all vertices in a set of vertices V is denoted $\text{ATB}(V)$, i.e., $\text{ATB}(V) = \bigcap_{v \in V} \text{ATB}(v)$. The graph induced by the vertices sharing a set of attributes X is denoted $\mathcal{G}[X]$, i.e., $\mathcal{G}[X] = \mathcal{G}[\{v \in V \mid X \subseteq \text{ATB}(v)\}]$, and $\mathcal{G}[X]$ is called the *subgraph of \mathcal{G} induced by the set of attributes X* .

Example 11 Consider the Boolean attributed graph presented in Figure 8. Only vertices C and D share attributes rock and pop, and then we have $\text{ATB}(\{C, D\}) = \{\text{rock}, \text{pop}\}$. As only vertices A and D share attributes pop and jazz, the subgraph induced by the set of attributes $\{\text{pop}, \text{jazz}\}$ and denoted $\mathcal{G}[\{\text{pop}, \text{jazz}\}]$ is $\mathcal{G}[\{A, D\}]$.

3.2 GLOBAL APPROACHES IN ATTRIBUTED GRAPHS

As mentioned in the introduction of this thesis, a common approach when analysing data is to perform a clustering. Clustering is the process of partitioning data objects into several groups with respect to a similarity measure, such that the objects within a group are similar to each other and dissimilar to data objects from other groups [55]. This task has been well studied in the context of binary relations and graphs without attributes. By taking into account both the graph topology and the attributes, the result of the clustering is expected to be more relevant than a clustering based solely on graph topology or attributes value. Let us give an overview of the clustering methods for attributed graphs.

Connected k-center

The *connected k-center* problem proposed in [24] is a first clustering approach in attributed graphs. It extends the k-center problem to attributed graphs. The original k-center problem consists in finding if there exists k cluster centroids such that the distance between all vertices within a cluster and its centroid is at most r . In the connected k-center problem, as proposed in [24], the distance between the vertices is based on the value of the attributes. To take into account graph topology, a connectivity constraint is added. This constraint requires that each subgraph induced by a cluster is a connected component.

To solve the connected k-center problem, the NetScan algorithm is proposed. This algorithm is similar to K-Medoids [43]. At first, k medoids are selected from the set of vertices. In a second step, the vertices are assigned to the closest cluster if they are within a distance r from the medoids and the connectivity constraint is satisfied. The third step consists in computing new medoids for each cluster. As long as

there is vertices not assigned to a cluster, the algorithm increases r (the maximum radius of a cluster), and goes on from the second step.

SA-Cluster: Structural/Attribute similarities clustering

In the work presented in [95] and extended in [96], the authors introduce another attributed graph clustering approach. One of the interests of their approach is based on the idea that attributes are not equally discriminative. For example, in a social network a gender attribute is less likely to discriminate a community than hobbies. Consequently, it might be interesting to associate more weight to some attributes while less importance is given to the others.

Here the domains of the attributes in the attributed graph are categorical, *i.e.*, not restricted to Boolean values. To take into account both the structure of the graph and the attributes, the authors propose to build an attribute augmented graph where each possible attribute values correspond to a new vertex. Such vertex is named attribute vertices. In such graph, the number of new attribute vertices is then the number of distinct values for all the attributes. A structure vertex v (a vertex of the original graph) is connected to an attribute vertex if v has the value corresponding to this attribute vertex.

It is supposed that the attributes have disjoint domains, eventually after an appropriated value renaming.

In this attributed augmented graph, the similarity between two structural vertices u and v is then computed using a random walk. The idea of a random walk is to start from a vertex and move to another vertex, possibly an attribute vertex, in the neighbourhood of v with a given probability. Two vertices are similar if it is possible to reach one from the other in a small expected number of steps. Clearly, with such technique, adding paths through attribute vertices between structural vertices allows to take into account attribute values when computing the vertices similarity. The clustering based on this similarity measure follows the K-Medoids clustering method [43].

The weight of the attributes are equal in the first iteration and updated at each iteration of the K-Medoids algorithm. If the vertices within a cluster tend to have the same value for an attribute, the weight associated to this attribute is increased, otherwise it is decreased. The attribute weight is then used to compute the probability of moving through this attribute vertex in the random walk. The extension proposed in [96] improves the runtime for the random walk computation by performing an incremental update of the random walk distance matrix.

Connected X Clusters

One drawback of the two previous approaches is that the user must provide K , the number of clusters while in general this number is not a priori known. To overcome this problem, the *Connected X Clusters* method has been proposed in [57] in order to perform a graph clustering using attribute values without specifying the number of clusters. The idea is to perform an initial clustering with a large number of clusters and then to merge the clusters by pairs until all clusters have been merged. The number of initial centroids is set to $\lceil k \cdot \ln \left(\frac{k}{-\ln(p)} \right) \rceil$ with $k = \lceil \frac{|G|}{m} \rceil$, m the minimal size of a cluster and p the probability that each true cluster is represented by at least one initial centroid. At each iteration, the algorithm considers all pairs of clusters (such that the two clusters shared at least a vertex) as a possible merge. Then, the merge that leads to the best clustering quality is performed. To assess

this quality, the authors propose a measure called the Joint Silhouette Coefficient extending the standard Silhouette Coefficient [44]. This measure takes into account both the edges of the graphs between the objects and their attribute values.

3.3 LOCAL PATTERNS IN ATTRIBUTED GRAPHS

Approaches targeted toward more local structure than a whole clustering have been proposed in the context of an attributed graph. For example, in [85] and [88], the authors propose matching techniques to find subgraphs within an attributed graph that best match a graph query. In this section, we will consider the mining of local pattern in attributed graphs.

3.3.1 Application specific approaches

The first results proposed regarding the discovery of local patterns in attributed graphs were application oriented. Indeed, the idea of using both the topology and the attributes has emerged early in molecular biology for the discovery of functional modules [38, 94, 86]. A functional modules is usually defined as a group of cellular components (*e.g.*, proteins) and their interactions such that this group can be associated to a specific biological function [39].

Another early application of attributed graphs was the analysis of social networks [21]. This work proposes to enrich the communities discovered in a social network with attributes. The patterns, corresponding to communities, are based on the union of cliques containing a given vertex. Once all patterns have been extracted, the corresponding communities are labelled with the attribute values considered to be representative using the attributes values associated to the objects forming the community. Note that this approach does not use the attribute information during the community detection process. Consequently, the communities discovered are not necessarily homogeneous with respect to the attributes values.

3.3.2 General frameworks

A few years after these applications, several works have tackled the problem of extending well studied graph local patterns to the context of attributed graphs in a more general way. This led to new classes of patterns: the cohesive patterns, the proximity patterns, the itemset-sharing subgraph set, and the structural correlation patterns.

Cohesive Patterns

In [58], the authors propose one of the first extension of dense graph mining to take into account the attributes. They use the term feature instead of attribute, however the concept remains identical. A feature vector graph is an attributed graph where the graph is undirected and unweighted and a categorical attribute corresponds to each feature.

Their patterns of interest are named *Cohesive Patterns*. Such a pattern is defined as a subgraph induced by a pseudo-clique (*i.e.*, connected subgraph having density above a threshold) and being homogeneous. The homogeneity corresponds to a subspace cohesion function $s : 2^V \times 2^A \times \mathbb{R} \mapsto \{\text{True}, \text{False}\}$ which returns True if and only if a set of

vertices associated to a subset of attributes and a real number used as a threshold is considered homogeneous. This function is not imposed by the framework, the only constraint is that the function must be anti-monotone. *I.e.*, with $V \subset V'$ two sets of vertices, $X \subset X'$ two sets of attributes, and x a real value, the function must hold $s(V, A, x) \Rightarrow s(V', A', x)$. For example, a subspace cohesion function might return True if and only if all vertices share the same value for the given set of attributes.

Proximity pattern

In [45] the authors propose the *proximity pattern* mining task. While cohesive patterns focus on dense subgraphs, proximity patterns are more related to frequent itemset mining. A proximity pattern is defined as a set of attributes such that (1) the vertices associated to these attributes are tightly connected in the graph and (2) they occur frequently. The interest of such pattern is to consider not only the attributes associated to a vertex (which would be similar to the traditional frequent itemset mining problem) but also the attributes associated to the vertices in the neighbourhood. It is sensible since in most contexts an object is influenced by its neighbourhood.

In order to find such patterns, the authors propose two models: the neighbour association model and the information propagation model. Since the authors experimentally show that the first model is not tractable in practice, we present only the information propagation model. The idea is to consider that the graph represents the reality at a given timestamp. After a while, the information (*i.e.*, the attributes) will propagate in the graph until it reach a stable state. The proximity patterns are then the frequent set of attributes in the attributed graph at the stable state.

Subspace Clustering

In [35], the authors propose a subspace clustering method allowing arbitrary shape of the clusters. Differently to the method proposed in [58], based on quasi-clique, they propose a density measure based on the size of the neighbourhood within the pattern.

To take into account both graph topology and attributes, they define two neighbourhoods for a vertex. One is the graph k -neighbourhood defined for a vertex as the set of vertices reachable in the graph by a path of size at most k . The other one is the ϵ -neighbourhood defined in the attribute space. More precisely, the ϵ -neighbourhood of a vertex v is the set of vertices such that the distance in the attribute space between v and these vertices is lesser than a given threshold ϵ , *i.e.*, $\{u \in V \mid \text{dist}(\text{ATB}(u), \text{ATB}(v)) \leq \epsilon\}$, with dist an arbitrary distance measure in the attribute space.

Itemset-sharing subgraph set

In [27] and [77] the authors propose the itemset-sharing subgraph set enumeration problem. An itemset-sharing subgraph set is defined as the collection of connected components in the subgraph induced by a non empty set of attributes, with a minimum bound, θ_S , on the number of vertices in the connected components. Note that this definition does not require the subgraph to be very dense. The *itemset-sharing subgraph set enumeration problem* consists then in finding all itemset-sharing subgraph sets such that (1) the set contains at least θ_F connected

While the term clustering is used in the name of the pattern, one can consider it as a local pattern since its validity is evaluated independently from the other patterns, a characteristic of local patterns as proposed in [73].

components and (2) the vertices forming the pattern share at least θ_1 attributes.

Structural Correlation Pattern

The work presented in [79] and extended in [80] and [54] introduces the structural correlation patterns. This class of patterns is based on a structural correlation measure for a set of attributes. For short, a set of attributes is considered to be structurally correlated if, in the subgraph induced by this set of attributes, a sufficient percentage of vertices are in dense subgraphs. More precisely, the structural correlation measure for a set of attributes is the ratio between the number of vertices being in a δ -quasi-clique in the subgraph induced by this set of attributes and the number of vertices in this subgraph. Using this structural correlation measure, it is possible to express how attribute values are related to the existence of dense components.

A *structural correlation pattern* is then defined as a δ -quasi-clique along with a set of attribute such that (1) the vertices share at least σ attributes, (2) the δ -quasi-clique contains at least \min_{size} vertices, and (3) the structural correlation measure corresponding to the set of attributes is above a threshold ϵ_{min} . This definition is similar to the cohesive patterns proposed in [58] since both are based on quasi-cliques being homogeneous with respect to the attributes. However, the structural correlation measure ensures that the attribute values associated to a pattern are strongly associated to the presence of δ -quasi-cliques.

CONCLUSION

In this state of the art we presented several data mining approaches for the analysis of attributed graphs. Since it is a relatively recent task in data mining we proposed to enlarge the scope to binary relations and graphs.

In the binary relation context, we considered groups of properties occurring frequently together. Three typical examples of such structures were presented: frequent itemsets, frequent closed itemsets and frequent error-tolerant closed itemsets. The objective was to present what has been considered as a group of properties associated to objects in the literature and the general concepts to extract such structures. In the graph setting, we introduced several graph measures used to characterize a graph. We also presented structures formed by groups of objects connected in a graph. Three examples of well studied structures were presented, namely, cliques, quasi-cliques, and k -clique percolated components. Our objective was to give an overview of common structures used to characterize dense or strongly connected subgraphs. Finally, we presented several data mining tasks in the context of attributed graphs with a wider scope, from clustering to local pattern extraction. The objective was to present existing approaches to find groups of vertices in attributed graphs. The interest of such structures is that their relevancy is evaluated not only from their topology within the graph but also from the values of the attributes associated to the vertices. Among the presented local patterns in attributed graphs, some are formed by a single dense subgraph being homogeneous (*e.g.*, cohesive patterns, proximity patterns) while the itemset-sharing subgraph set is formed by a collection of subgraphs not required to be dense.

The approach proposed in this thesis is at the boundaries of these previous works. More precisely, we propose to study structures formed by *collections* of homogeneous and *dense* subgraphs. The discovery of such patterns will be studied in the next part with the objectives of being efficient, tolerant to missing values, and to allow the experts to specify the structures of interest by mean of constraints.

Part III

A PATTERN AS A COLLECTION OF
SUBGRAPHS

INTRODUCTION

The analysis of attributed graphs opened the possibility to take into account more information than just simply the topology of the graph. Up to now, using both graph topology and the attributes associated to vertices has allowed to propose new classes of patterns. The relevancy of these patterns is evaluated not only from the structure within the graph, but also from some measure with respects to the attributes. The fact that the relevancy of a pattern is evaluated by two measures leads to another interesting property. Indeed, when two measures of interest are available, it is possible to group the relevant patterns with respect to the first measure when they are similar with respect to the second measure. In our work we will develop this idea, to find not a single group of vertices but structures that are organized in several groups. For example, consider a measure of density and a measure of homogeneity. It is possible to look for a set of dense groups whose union is homogeneous. In the context of a social network, that would be several communities¹ sharing common interests instead of a single community having similar interests. From now on, we will consider that a collection of subgraphs is homogeneous when the union of the vertices forming the subgraphs is homogeneous. In the next two sections, we consider the Boolean attributed graph context, and propose two pattern definitions in order to find collections of subgraphs which are both dense and homogeneous: the Maximal Homogeneous Clique Sets and the Collections of Homogeneous k -clique Percolated components.

The Maximal Homogeneous Clique Sets

The first patterns we introduce are the *Maximal Homogeneous Clique Sets* (MHCS for short). The term *maximal* refers to the fact that the patterns are the most general. *Homogeneous* refers to the homogeneity of the vertices within the pattern. The term *clique* refers to the density constraint on the vertices, and the term *sets* refers to the fact that we seek a collection of subgraphs instead of a single subgraph. More precisely, a MHCS is a group of cliques satisfying constraints on the number of separated cliques, the size of the cliques and the number of attributes shared by all vertices. The constraints on the minimal size and the minimum number of shared attributes are similar to the ones proposed in [27] for the Itemset-Sharing Subgraph problem (see Section 3.3.2).

We present an example of such pattern in the Boolean attributed graph presented Figure 9, already introduced page 4 but reproduced here to ease the reading. This dataset represents a set of individuals and relationships between them. Each person corresponds to a vertex denoted by a capital letter, and the relationships are represented by edges. Such relationships might represent friendship, geographic closeness, or being member of the same organization. Moreover, the musical tastes (*e.g.*, Rock, Pop) of each person is encoded by means of Boolean attributes. A person associated to an attribute having the value

1. If we adopt a density-based definition of communities. See [19] for an overview of the different definitions of communities.

True represents the fact that this person enjoy the corresponding style of music. In Figures 9 and 10, only attributes having value True are represented.

In this setting, consider the MHCSs formed by a collection of at least two 4-maximal cliques sharing at least two attributes. The collection of set of vertices $\{ABCD, EGH, M, NOPQR\}$ is such a MHCS. Indeed, it contains two cliques with at least four vertices (ABCD and NOPQR) and all the vertices share the attributes rock and folk. The vertex F, which might be used to build the 4-maximal clique EFGH, is not in the pattern since it does not share these two attributes with the other vertices. Note that the clique EGH and the single vertex clique M are also in the pattern even if they have less than four vertices. While the κ -maximal cliques form the core part of the pattern, the smaller cliques point out vertices sharing the same two attributes but being more isolated.

As a comparison with other local patterns in attributed graphs, ABCD or NOPQR might be considered as maximal cohesive patterns [58] or structural correlation patterns [79] using reasonable parameters. Indeed, all vertices in ABCD share attributes rock, folk and jazz. Considering NOPQR the vertices share attributes rock, folk, and pop. Note that even though both ABCD and NOPQR share attributes rock and folk, these two patterns would not be considered as related in the output when looking for cohesive patterns or structural correlation patterns.

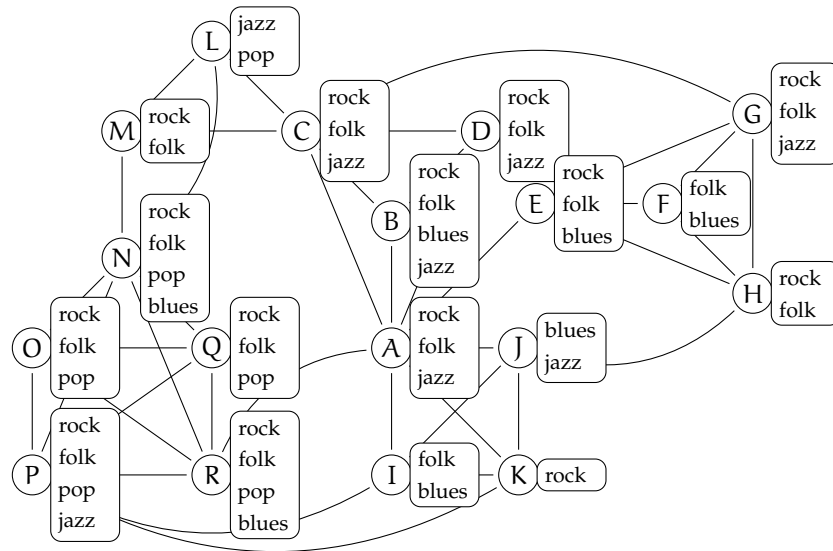


Figure 9.: Example of Boolean attributed graph representing a social network (same as Figure 1).

Extension to tolerate missing edges in the groups

We have seen in the state of the art that local patterns approaches have tend to evolve to take into account missing values in the data. For example closed frequent patterns gave rise to frequent closed error tolerant itemsets, and cliques to quasi-cliques. Following the same trend, we extend the MHCS approach to allow missing edges within the cliques forming the pattern.

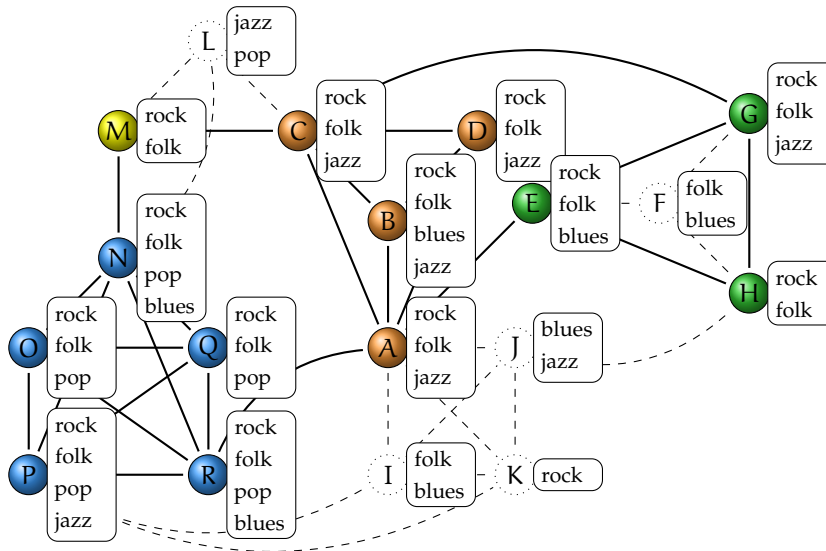


Figure 10.: Example of MHCS pattern in the Boolean attributed graph presented in Figure 9.

Being able to find dense but not necessarily complete subgraphs (tolerating missing edges) allows to (1) avoid finding very similar subgraphs and (2) do not “miss” groups due to the presence of a few missing edges.

Two methods are commonly used to find dense subgraphs with missing edges: quasi-cliques (see Section 2.2.2), and k -PC, (k -clique percolated components, see Section 2.2.3). Compared to the definitions of quasi-cliques, the definition of k -PCs ensures that two complete subgraphs sharing many common vertices are merged in the same k -PC (*i.e.*, in the same group of vertices).

So, we propose a definition of patterns based on k -PCs where a pattern is a *Collection of Homogeneous k -clique Percolated component* (CoHoP for short). The term *collection* refers to the fact that we want several subgraphs, *homogeneous* refers to the attributes shared by the vertices forming the pattern and *k -clique percolated component* refers to the topology of the subgraphs.

Outline

In the next section (Section 4.1), we give a constraint based definition of the MHCS patterns. The interest of the constraints is illustrated with several examples. Then, we refine our first pattern definition to remove redundant patterns.

In Section 4.2, we propose a correct algorithm based on subgraph enumeration to extract all patterns. From a naive approach which is intractable in practice, we improve the efficiency by using several pruning techniques. A formal proof of the correctness is given for each pruning.

Experiments are presented on bibliographic data in Section 4.3. We give several examples of MHCSs and illustrate how they can be used to support decisions regarding the selection of a research supervisor, the selection of article reviewers, and the elaboration of scientific collaborations. We also perform a quantitative evaluation of our algorithm

Removing an edge in a k -maximal clique leads to two $(k - 1)$ -maximal cliques having $k - 2$ vertices in common.

showing that the extraction remains possible for large attributed graphs. The runtime evolution is also studied with respect to different graph structures and user parameters using synthetic graphs. Finally, we study the impact on runtime of each pruning techniques compared to a baseline algorithm.

Then, we extend the MHCS definition to find collections of homogeneous subgraphs with missing edges, namely k -PCs (Section 5.1). We propose an extraction algorithm based on subgraph enumeration using several pruning techniques (Section 5.2). We perform the same type of experiments as for the MHCSs, considering also a network of scientific collaborations and synthetic datasets (Section 5.3).

MINING COLLECTIONS OF CLIQUES HAVING HOMOGENEOUS VERTICES

4.1 PATTERN DEFINITION

In this section, we first recall the Boolean attributed graph setting. Then we propose a constraint based definition of the Homogeneous Clique Set (HCS) pattern and illustrate the interest of the constraints. Finally, we introduce the Maximal Homogeneous Clique Sets (MHCS) patterns.

Before giving a formal definition of our pattern of interest, let us recall the Boolean attributed graph setting as presented in Section 3 of the state of the art. A Boolean attributed graph is denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \text{ATB})$ where \mathcal{V} is the set of vertices, \mathcal{E} is the set of edges, \mathcal{A} is the set of Boolean attributes, and $\text{ATB} : \mathcal{V} \rightarrow 2^{\mathcal{A}}$ is the function returning for a vertex the set of attributes having value True. We denote $\mathcal{G}[V]$ the subgraph of \mathcal{G} induced by the set of vertices $V \subseteq \mathcal{V}$ and $\mathcal{G}[A]$ the subgraph induced by the set of vertices having value True for all attributes in $A \subseteq \mathcal{A}$.

For notational convenience, we will also define the next two functions.

Definition 14 (Functions `VERT` and `CATB`) Let x be an attribute. The function $\text{VERT}(x) = \{v \in \mathcal{V} \mid x \in \text{ATB}(v)\}$ is the set of vertices having value True for the attribute x . Let M be a collection of sets of vertices. Then, $\text{CATB}(M) = \bigcap_{V \in M} (\bigcap_{v \in V} \text{ATB}(v))$ is the set of attributes shared by all vertices in M .

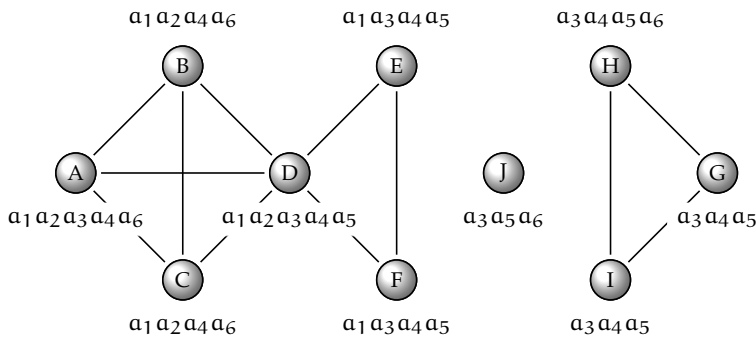


Figure 11.: Example of Boolean attributed graph. Vertices are identified by capital letters and attributes are identified by attributes a_i , $i \in \{1, \dots, 6\}$.

For the sake of simplicity in the examples, a set of vertices is simply denoted by the sequence of letters corresponding to the vertices.

Example 12 In the example of Boolean attributed graph presented Figure 11, $\text{VERT}(a_6) = \text{ABCJH}$. Considering $M = \{\text{DEF}, \text{A}, \text{GHI}\}$ a collection of set of vertices, $\text{CATB}(M) = \{a_3, a_4\}$.

In a Boolean attributed graph dataset, our goal is to find collections of homogeneous cliques. Each pattern is a collection of set of vertices, such that:

- the vertices are homogeneous in the sense that they share some attributes (attributes having value True for all vertices);
- the pattern contains several large groups of strongly connected vertices (the other groups in the pattern can be small and even reduced to a single vertex);
- the cliques in the pattern are separated (two groups in the pattern cannot be merge trivially to form another group).

These patterns, termed *homogeneous clique sets* (HCS) are defined as follows:

Definition 15 (Homogeneous Clique Set) Let κ , α , and γ be three strictly positive integers, and \mathcal{G} be a Boolean attributed graph. A collection of cliques $M = \{C_1, \dots, C_n\} \subseteq \mathcal{C}(\mathcal{G})$ is an Homogeneous Clique Set (HCS) if and only if the three following constraints $C_\alpha^{\text{hom}}\text{o}$, $C_{\gamma,\kappa}^{\text{clique}}$ and C^{sep} are satisfied:

- $C_\alpha^{\text{hom}}\text{o}(M) \equiv |\text{CATB}(M)| \geq \alpha$, i.e., all vertices share at least α attributes;
- $C_{\gamma,\kappa}^{\text{clique}}(M) \equiv |\{C \in M \mid |C| \geq \kappa\}| \geq \gamma$, i.e., M contains at least γ cliques of size at least κ ;
- $C^{\text{sep}}(M) \equiv M = \mathcal{C}_{\text{max}}(\mathcal{G}[\bigcup_{C \in M}])$, i.e., M is the collection of maximal cliques in the subgraph induced by all the vertices appearing in M .

Before presenting the interest of each constraint, let us give an example of a HCS using the toy dataset presented Figure 11.

Example 13 (Homogeneous Clique Set) Consider the set of cliques $M = \{ABCD, DEF\}$. It satisfies the constraint $C_2^{\text{hom}}\text{o}$ since all vertices are associated to attributes α_1 and α_4 . It also satisfies $C_{2,3}^{\text{clique}}$ as the set contains two cliques having at least three vertices. Finally, as cliques are maximal in $\mathcal{G}[ABCDEF]$, then P also satisfies C^{sep} , and thus is a HCS. Note that since cliques can overlap, vertex D is in two cliques.

We will now illustrate the interest of each constraint.

4.1.1 The homogeneity constraint: $C_\alpha^{\text{hom}}\text{o}$

This constraint ensures that all vertices in a pattern share a set of attributes. This set of attributes ensures the homogeneity of the pattern. Since α is the minimal number of shared attributes, the higher is α , the more homogeneous the patterns will be.

The homogeneity threshold is absolute, however one can obtain a relative threshold by dividing α by $|A|$.

Example 14 (Constraint $C_\alpha^{\text{hom}}\text{o}$) Consider the attributed graph presented Figure 11. As the set of cliques $\{ABCD, DEF, GHI\}$ is built on vertices associated to attributes α_4 , this set satisfies $C_1^{\text{hom}}\text{o}$. More stringent constraints $C_\alpha^{\text{hom}}\text{o}$ can be used to focus on sets of cliques that are more homogeneous, i.e., that share more attributes. For instance, the set of cliques $\{DEF, GHI\}$ satisfies $C_3^{\text{hom}}\text{o}$ as all vertices are associated to attributes α_3 , α_4 , and α_5 , while this constraint is not satisfied by $\{ABCD, DEF, GHI\}$.

4.1.2 The topology constraint: $C_{\gamma,\kappa}^{\text{clique}}$

This constraint is used to avoid small patterns, i.e., collections of small subgraphs or collections of few large subgraphs. The analyst might want to avoid such patterns as they usually do not provide valuable information. The parameter γ define the minimal number of

cliques in the pattern having a size of at least κ vertices. Note that a pattern might contain cliques with less than κ vertices as long as there is at least γ cliques of size κ . The cliques of size at least κ have a straightforward interest as forming the core part of the pattern (large groups of strongly connected vertices). The cliques of size less than κ are kept in the pattern since they point out vertices sharing the same attributes but being more isolated. In the experiments, we will present several patterns illustrating the interest of these isolated groups (see Section 4.3). However, if needed the cliques with less than κ vertices can be removed using a simple post-processing.

Example 15 (Constraint $C_{\gamma,\kappa}^{\text{clique}}$) *The vertices A,B,C,H, and J share attributes a_6 . Using these vertices we can build the HCS $\{ABC, H, J\}$ satisfying $C_{1,3}^{\text{clique}}$. On the dataset of Figure 11, a more stringent constraint would be $C_{2,3}^{\text{clique}}$, to ask for at least 2 groups of size 3. Then $\{ABC, H, J\}$ is no longer retrieved as a HCS, while we still obtain, for instance, $\{A, DEF, GHI\}$ (sharing attributes a_3 and a_4).*

4.1.3 The separation constraint: C^{sep}

This constraint is needed to avoid that a single large clique could be counted as a collection of smaller cliques, since it could hardly be considered as an interesting collection cliques. The C^{sep} constraint requires that the union of two cliques within a pattern is not a clique itself. The following example illustrates this case using the graph presented Figure 11.

Note that this does not require that the cliques are maximal cliques of the whole graph \mathcal{G}

Example 16 (Constraint C^{sep}) *Consider the vertices A, B, C and D sharing attributes a_2 and a_4 . Using these vertices, we can build a set of four cliques of size three $\{ABC, ABD, ACD, BCD\}$ that would satisfy the constraint $C_{4,3}^{\text{clique}}$. However, since ABCD is itself a clique, the collection $\{ABC, ABD, ACD, BCD\}$ does not satisfy C^{sep} .*

4.1.4 Reducing the collection of patterns

A common issue when extracting patterns is to provide small and easy to browse collections [14]. By definition, a set of HCSs can contain redundant patterns, in the sense that, when we know the parameters κ and γ , some patterns can be directly derived from others. Indeed, consider any set of cliques M' obtained by removing some vertices from a HCS M . If M' satisfies $C_{\gamma,\kappa}^{\text{clique}}$ and C^{sep} then M' is also a HCS. This is illustrated by the following example.

Example 17 (Redundant patterns) *In Figure 11, using vertices sharing attributes a_1 and a_4 , we can build a HCS $\{ABCD, DEF\}$ satisfying the constraints C_2^{homo} , $C_{2,3}^{\text{clique}}$, and C^{sep} . We can also build four other HCSs satisfying the same constraints by removing any vertex from the clique ABCD, e.g., $\{ACD, DEF\}$.*

Since the number of HCSs formed by cliques which are subsets of cliques in another HCS can be large, such redundant patterns are discarded by focusing on *maximal* HCSs only.

Definition 16 (Maximal Homogeneous Clique Set) *A Maximal Homogeneous Clique Set (MHCS) is a HCS which is maximal with respect to the*

partial order \preceq defined as follows. Given M_1 and M_2 two HCSs, $M_1 \preceq M_2$ if and only if for all $C_1 \in M_1$ there exists $C_2 \in M_2$ such that $C_1 \subseteq C_2$.

Notice that antisymmetry does not hold for \preceq in an arbitrary collection of sets, and thus \preceq is not necessarily a partial ordering relation. However, in the case of a collection of HCSs, the relation \preceq is a partial order as stated by the following property.

Property 1 *On a collection of HCSs, the relation \preceq is a partial order.*

Proof 1 *The relation is trivially reflexive and transitive. To show antisymmetry, consider M_1 and M_2 two HCSs such that $M_1 \preceq M_2$ and $M_2 \preceq M_1$. Suppose that $M_1 \neq M_2$, then there exists C in M_1 that is different from all sets in M_2 . And since $M_1 \preceq M_2$, there exists C' in M_2 such that $C \subseteq C'$. As, $M_2 \preceq M_1$, there exists C'' in M_1 such that $C' \subseteq C''$. So, $C \subseteq C''$, but this cannot hold since by definition of a HCS M_1 must satisfy C^{sep} . Thus $M_1 = M_2$.*

Example 18 (Maximal Homogeneous Clique Set) *The two sets of cliques $\{ABCD, DEF\}$ and $\{A, DEF, GHI\}$, sharing respectively attributes $\{a_1, a_4\}$ and $\{a_3, a_4\}$, are MHCSs in the graph depicted Figure 11, for constraints C_2^{homo} and $C_{2,3}^{\text{clique}}$. Satisfying the same constraints, there are other HCSs that are not maximal ones, as for instance $\{ABC, DEF\}$ and $\{DEF, GHI\}$.*

4.2 FINDING ALL MAXIMAL HOMOGENEOUS CLIQUE SETS

In this section we present a sound and complete algorithm to find all MHCSs in a given dataset. For the sake of clarity, three versions of the algorithm are proposed. A naive generate and test approach is first presented and several pruning techniques are then incorporated in two successive versions. The final version is given as Algorithm 8.

4.2.1 Algorithm generate and test

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \text{ATB})$ be a Boolean attributed graph. The algorithm enumerates in a depth first way subgraphs of \mathcal{G} together with the sets of attributes shared by the vertices of these subgraphs. The extraction of all MHCSs is done in two steps. The main step consists in the extraction of all MHCSs together with some non maximal HCSs (`ExploreSubgraphs1`, Algorithm 4), and then a second step filters out these non maximal patterns (`ExtractMHCS`, Algorithm 5).

Before presenting the naive algorithm, let us define the function `VERT` which is an extension of `VERT` for a set of attributes instead of a single attribute.

Definition 17 `VERT`. *The function $\text{VERT} : 2^{\mathcal{A}} \mapsto 2^{\mathcal{V}}$ is defined as $\text{VERT}(A) = \bigcap_{a \in A} \text{VERT}(a)$. It maps a set of attributes A to the set of vertices associated to all the attributes in A .*

Example 19 *In the attributed graph presented Figure 11, $\text{VERT}(a_1 a_4 a_5) = \text{DEF}$ since all these vertices and only them are associated to attributes a_1 , a_4 , and a_5 .*

In the recursive function `ExploreSubgraphs1`, the first parameter \mathcal{G}_e is the current enumerated (sub-)graph to be tested, \mathcal{A}_s is a set of attributes already known to be shared by all vertices of \mathcal{G}_e , \mathcal{V}_r is the set of vertices

remaining in \mathcal{G}_e , and \mathcal{A}_c is the set of candidate attributes that remain under consideration to find attributes shared by \mathcal{G}_e or by subgraphs of \mathcal{G}_e . The initial call to `ExploreSubgraphs1` is made in `ExtractMHCS` with parameter \mathcal{G}_e set to the whole graph, $\mathcal{A}_s = \emptyset$, $\mathcal{V}_r = \mathcal{V}$, and $\mathcal{A}_c = \mathcal{A}$.

Starting from the whole graph and with $\mathcal{A}_c = \mathcal{A}$, Algorithm 4 enumerates all subsets $\mathcal{A}_s \subseteq \mathcal{A}$ (all sets of attributes that can be shared) together with the subgraph \mathcal{G}_e of \mathcal{G} composed of the vertices associated to all the attributes in \mathcal{A}_s . For each of the enumerated subgraph, the algorithm first checks on line 2 if the set of maximal cliques of the subgraph is considered to be a HCS (*i.e.*, contains at least γ cliques having κ vertices and if the vertices are already known to share at least α attributes). If so, the pattern is added to the result HCSs. In this preliminary version of the algorithm no pruning is performed here, and the enumeration always goes on (lines 4 to 8).

This enumeration uses the set \mathcal{A}_c containing the attributes that remain candidates as attributes shared by the current graph or its subgraphs. While \mathcal{A}_c is not empty, an attribute y is picked and removed from \mathcal{A}_c , and a set of vertices \mathcal{V}'_r is built by restricting the current set of vertices \mathcal{V}_r to the vertices associated to attribute y . The set of attributes known to be shared by the vertices in \mathcal{V}'_r is then simply $\mathcal{A}'_s = \mathcal{A}_s \cup \{y\}$. Then function `ExploreSubgraphs1` is called recursively on $\mathcal{G}_e[\mathcal{V}'_r]$ (the subgraph of \mathcal{G}_e induced by the set of vertices \mathcal{V}'_r).

Algorithm 4: `ExploreSubgraphs1`

Input: $\mathcal{G}_e, \mathcal{A}_s, \mathcal{V}_r, \mathcal{A}_c$
1 $\text{HCSs} \leftarrow \emptyset$
2 **if** $|\mathcal{A}_s| \geq \alpha$ **and** $|\{C \in \mathcal{C}_{\max}(\mathcal{G}_e) \mid |C| \geq \kappa\}| \geq \gamma$ **then**
3 $\text{HCSs} \leftarrow \{\mathcal{C}_{\max}(\mathcal{G}_e)\}$
4 **while** $\mathcal{A}_c \neq \emptyset$ **do**
5 Pick and remove an element y from \mathcal{A}_c
6 $\mathcal{V}'_r \leftarrow \mathcal{V}_r \cap \text{VERT}(y)$
7 $\mathcal{A}'_s \leftarrow \mathcal{A}_s \cup \{y\}$
8 $\text{HCSs} \leftarrow \text{HCSs} \cup \text{ExploreSubgraphs1}(\mathcal{G}_e[\mathcal{V}'_r], \mathcal{A}'_s, \mathcal{V}'_r, \mathcal{A}_c)$
9 **return** HCSs

Algorithm 5: `ExtractMHCS`

Input: \mathcal{G} , an attributed graph
1 $\text{HCSs} \leftarrow \text{ExploreSubgraphs1}(\mathcal{G}, \emptyset, \mathcal{V}, \mathcal{A})$
2 **output** `RemoveNonMaximalHCSs`(HCSs)

In order to illustrate the general recursive scheme, and to introduce the pruning techniques, let us define the underlying enumeration tree.

Definition 18 (Node in the enumeration tree) *A node i in the enumeration tree represents a recursive call to `ExploreSubgraphs1` in Algorithm 4. A total order $<$ is defined on the nodes, based on the calling sequence during an execution, *i.e.*, given i and j two nodes in the enumeration tree, $i < j$ if and only if the call to `ExploreSubgraphs1` for node i is done before the call to `ExploreSubgraphs1` for node j . A node i is the father of a node j and j is a child of i if and only if the call to `ExploreSubgraphs1` corresponding to j is done line 8 in the call corresponding to i . The ancestor relation is simply*

the transitive closure of relation *father*. Parameters of the call corresponding to node i are denoted using i as superscript: \mathcal{G}_e^i , \mathcal{A}_c^i , \mathcal{V}_r^i , and \mathcal{A}_c^i .

Example 20 (Enumeration tree) Figure 12 depicts a part of an enumeration tree that could be obtained on the graph presented Figure 11. The labels on the branches correspond to the attribute y picked to generate the calls. The nodes are identified by arbitrary increasing integers reflecting the calling sequence (order $<$ on nodes). For instance node 0 is the initial call to *ExploreSubgraphs1* (the ancestor of all nodes), node 33 is a call performed before the call corresponding to node 49, and node 56 is a child of node 49. In Figure 12, for the nodes at depth 1 and the nodes on the left-most branch, two input parameters of the call are indicated: \mathcal{V}_r (vertices in the current graph), and \mathcal{A}_c (remaining candidate attributes). Each possible subset of attributes is enumerated together with the subgraph induced by the vertices sharing these attributes. The other subgraphs are not enumerated, e.g., the subgraph induced by the vertices $\{B, C, D, E, F, G, H, I, J\}$.

To establish the completeness of the enumeration, we need the properties stated by the following lemmas.

Lemma 1 For all MHCS M in a Boolean attributed graph \mathcal{G} we have $M = \mathcal{C}_{\max}(\mathcal{G}[\mathbf{CATB}(M)])$.

Proof 2 Let M be a MHCS in a Boolean attributed graph \mathcal{G} . By definition $M = \mathcal{C}_{\max}(\mathcal{G}[\bigcup_{C \in M} C])$. Let $M' = \mathcal{C}_{\max}(\mathcal{G}[\mathbf{CATB}(M)])$ and suppose that $M \neq M'$. Then M' satisfies \mathcal{C}^{sep} by construction, and since the set of attributes shared by the vertices in M' is $\mathbf{CATB}(M)$ and M satisfies $\mathcal{C}_\alpha^{\text{homo}}$, then M' satisfies $\mathcal{C}_\alpha^{\text{homo}}$. By definition of functions \mathbf{VERT} and \mathbf{CATB} , $\bigcup_{C \in M} C \subseteq \mathbf{VERT}(\mathbf{CATB}(M))$, so $\mathcal{G}[\bigcup_{C \in M} C]$ is an induced subgraph of $\mathcal{G}[\mathbf{CATB}(M)]$. So, M' satisfies $\mathcal{C}_{\gamma, \kappa}^{\text{clique}}$ since M satisfies $\mathcal{C}_{\gamma, \kappa}^{\text{clique}}$, and then M' is a HCS such that $M \preceq M'$. This is not possible since M is a MHCS. Whence, $M = M'$.

Since for a node i in the enumeration tree, Algorithm 4 enumerates all subgraphs $\mathcal{G}_e^i[\mathbf{VERT}(X)]$ such that $X \subseteq \mathcal{A}_c^i$, then the following property is straightforward.

Lemma 2 For all node i in the enumeration tree, if there exists $A \subseteq \mathcal{A}_c^i$ and M a MHCS such that $M = \mathcal{C}_{\max}(\mathcal{G}_e^i[\mathbf{VERT}(A)])$, then M is obtained in the subtree rooted at node i .

The following theorems state the correctness of the extraction process.

Theorem 1 Algorithm 4 outputs all MHCSs and only HCSs.

Proof 3 Let \mathcal{G} be an attributed graph and M a MHCS in \mathcal{G} . Consider the root ($i = 0$) of the enumeration tree explored by Algorithm 4, then we have $\mathcal{G}_e^0 = \mathcal{G}$ and $\mathcal{A}_c^0 = \mathcal{A}$. Let $A = \mathbf{CATB}(M)$, then we know by Lemma 1 that $M = \mathcal{C}_{\max}(\mathcal{G}[\mathbf{VERT}(A)])$. And finally, Lemma 2 ensures that M is obtained in the subtree rooted at $i = 0$.

In addition, since the Algorithm 5 filters out non maximal HCS, we have the following property.

Theorem 2 Algorithm 5 is correct.

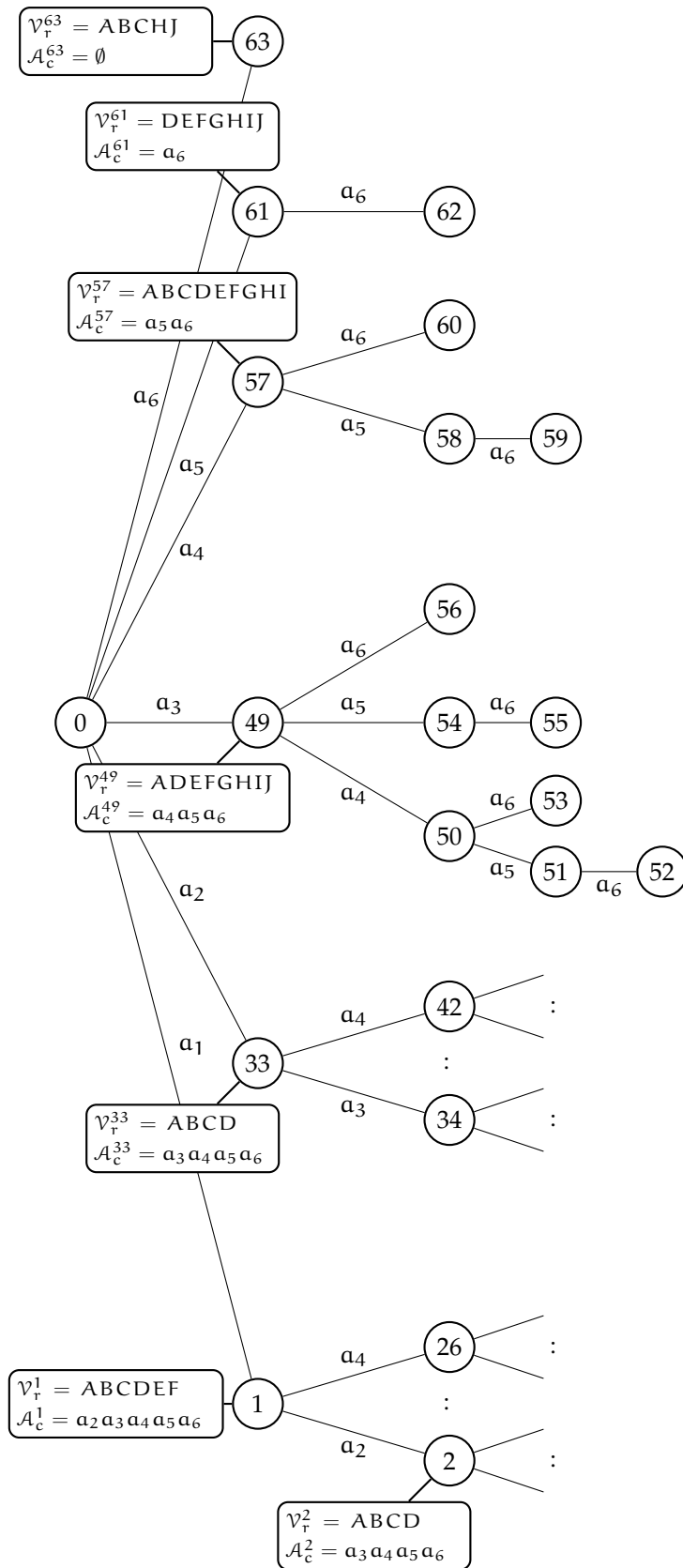


Figure 12.: Enumeration tree corresponding to the attributed graph presented Figure 11.

4.2.2 Enumeration tree pruning techniques

Five safe pruning techniques are used. Algorithm 6 extends Algorithm 4 and introduced the first four pruning techniques. For the sake of clarity, the fifth pruning technique is described in Algorithm 8 as an extension of Algorithm 6.

For each technique, we discuss its safety, and illustrate the corresponding pruning on an enumeration tree presented Figure 13 and obtained for the dataset of Figure 11 under constraints $\mathcal{C}_2^{\text{hom}}_o$ and $\mathcal{C}_{2,3}^{\text{clique}}$.

PRUNING 1 (Algorithm 6 line 2): This pruning checks that there is at least γ maximal cliques with at least κ vertices in \mathcal{G}_e^i . If it is not the case no subgraph of \mathcal{G}_e^i can satisfy $\mathcal{C}_{\gamma,\kappa}^{\text{clique}}$, and thus the subtree rooted at i can be pruned. Considering node 33, we have $\mathcal{V}_r^{33} = \{A, B, C, D\}$. Since \mathcal{G}_e^{33} (the subgraph induced by $\{A, B, C, D\}$) does not contain at least two maximal cliques with three vertices, the subtree rooted at node 33 is pruned.

PRUNING 2 (Algorithm 6 lines 3 and 4): Attributes from \mathcal{A}_c^i shared by all vertices in \mathcal{V}_r^i are added to \mathcal{A}_s^i and removed from \mathcal{A}_c^i . This prunes the tree by avoiding to pick these attributes to create new children of node i . Removing these attributes from \mathcal{A}_c^i does not change the collection of different subgraphs enumerated in the tree, since if we pick such an attribute y to create a child we have $\mathcal{G}_e^i[\mathcal{V}_r^i \cap \text{VERT}(y)]$ that is equal to \mathcal{G}_e^i itself. Finally, since these attributes are added to \mathcal{A}_s^i , then $|\mathcal{A}_s^i|$ is the correct total number of attributes shared by all vertices of the graph at enumeration node i . Considering node 1, we have $\mathcal{A}_c^1 = \{a_2, a_3, a_4, a_5, a_6\}$ and $\mathcal{V}_r^1 = \{A, B, C, D, E, F\}$. Since $\mathcal{V}_r^1 \subseteq \text{vert}(a_4)$, attribute a_4 is added to \mathcal{A}_s^1 and removed from \mathcal{A}_c^1 (lines 3 and 4). And thus, the branch rooted at node 1 corresponding to attribute a_4 is pruned.

PRUNING 3 (Algorithm 6 line 7): If the set of attributes shared by all vertices has a cardinality greater than or equal to α (line 5), and as the current graph already satisfies to the test of line 2, then it contains a HCS $M = \{\mathcal{C}_{\max}(\mathcal{G}_e)\}$ that is collected in the result line 6. In this case, since all attributes shared by all vertices of \mathcal{G}_e^i has been removed from \mathcal{A}_c^i (line 4) then all graphs in the subtree rooted at i will contain strictly less vertices than \mathcal{G}_e^i and thus cannot lead to HCS M' such that $M \preceq M'$. Whence, the subtree rooted at i does not contain a maximal HCS and can be pruned (line 7). Considering node 54, we have $\mathcal{A}_s^{54} = \{a_3, a_4\}$ and $\mathcal{V}_r^{54} = \{D, E, F, G, H, I, J\}$. Since the set of cliques $\{DEF, GHI, J\}$ is a HCS, no subgraph of \mathcal{G}_e^{54} can contains a MHCS, thus the subtree rooted at node 54 is pruned.

The fourth pruning technique requires the following property.

Lemma 3 *Given a Boolean attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \text{ATB})$ with $|\mathcal{V}| < \max\left(\kappa, \lceil \frac{3 \log(\gamma)}{\log(3)} \rceil\right)$, a set of cliques of \mathcal{G} cannot satisfy $\mathcal{C}_{\gamma,\kappa}^{\text{clique}} \wedge \mathcal{C}^{\text{sep}}$.*

Proof 4 *Since a HCS satisfies $\mathcal{C}_{\gamma,\kappa}^{\text{clique}}$, it contains at least one clique with κ vertices, and thus the corresponding graph has at least κ vertices. Moreover [56] demonstrates that the maximum number of maximal cliques in a graph*

with n vertices is $3^{n/3}$, and so, a graph must have at least $\lceil \frac{3 \log(\gamma)}{\log(3)} \rceil$ vertices to contain γ maximal cliques.

PRUNING 4 (Algorithm 6 lines 8 and 9): Attributes and vertices which are not valid candidates to build a MHCS are removed from \mathcal{A}_c^i and \mathcal{V}_r^i :

1. All current vertices share $|\mathcal{A}_s^i|$ attributes, thus to be part of a HCS satisfying $\mathcal{C}_\alpha^{\text{homo}}$, a remaining vertex must have at least $\alpha - |\mathcal{A}_s^i|$ more attributes. Moreover, these $\alpha - |\mathcal{A}_s^i|$ attributes must be in \mathcal{A}_c^i (the candidate attributes to be shared by the vertices of the graphs in the enumeration subtree rooted at i). Any vertex that does not satisfy to this condition (line 8) can be safely removed.
2. As stated by Lemma 3, to contain a HCS satisfying $\mathcal{C}_{\gamma, \kappa}^{\text{clique}} \wedge \mathcal{C}^{\text{sep}}$ a graph must have a number of vertices greater than or equal to $\max\left(\kappa, \lceil \frac{3 \log(\gamma)}{\log(3)} \rceil\right)$. Then, an attribute associated in \mathcal{G}_e^i to less vertices than this lower bound cannot be an attribute shared by the vertices of a HCS found in the subtree rooted at i . Thus it can be removed from the remaining candidate attributes \mathcal{A}_c^i (line 9).

To illustrate these reductions of \mathcal{A}_c^i and \mathcal{V}_r^i , let us consider node 61. We have $\mathcal{A}_c^{61} = \{a_6\}$, $\mathcal{V}_r^{61} = \{D, E, F, G, H, I, J\}$, and $\mathcal{A}_s^{61} = \{a_5\}$. Since no attribute in \mathcal{A}_c^{61} is shared by all vertices in \mathcal{V}_r^{61} , then \mathcal{A}_s^{61} and \mathcal{A}_c^{61} are not modified by lines 3 and 4. Since $\alpha = 2$ and $|\mathcal{A}_s| = 1$, then line 8 keeps in \mathcal{V}_r^{61} only vertices associated to at least one attribute in \mathcal{A}_c^{61} . Thus, \mathcal{V}_r^{61} is reduced to $\{H, J\}$. On line 9, for $\gamma = 2$ and $\kappa = 3$ we have $\max\left(\kappa, \lceil \frac{3 \log(\gamma)}{\log(3)} \rceil\right) = 3$, and since $|\text{vert}(a_6) \cap \mathcal{V}_r^{61}| = 2$ (i.e., attribute a_6 appears only on two vertices in \mathcal{V}_r^{61}) then a_6 is removed from \mathcal{A}_c^{61} . So, the branch rooted at node 61 corresponding to attribute a_6 is pruned.

Algorithm 6: ExploreSubgraphs2

Input: $\mathcal{G}_e, \mathcal{A}_s, \mathcal{V}_r, \mathcal{A}_c$

```

1 HCSs  $\leftarrow \emptyset$ 
2 if  $\{C \in \mathcal{C}_{\max}(\mathcal{G}_e) \mid |C| \geq \kappa\} \geq \gamma$  then // Pruning 1
3    $S \leftarrow \{l \in \mathcal{A}_c \mid \mathcal{V}_r \subseteq \text{VERT}(l)\}$  // Pruning 2
4    $\mathcal{A}_c \leftarrow \mathcal{A}_c \setminus S$ ;  $\mathcal{A}_s \leftarrow \mathcal{A}_s \cup S$  // Pruning 2
5   if  $|\mathcal{A}_s| \geq \alpha$  then
6     HCSs  $\leftarrow \{\mathcal{C}_{\max}(\mathcal{G}_e)\}$ 
7   else // Pruning 3
8      $\mathcal{V}_r \leftarrow \{v \in \mathcal{V}_r \mid |\text{ATB}(v) \cap \mathcal{A}_c| \geq \alpha - |\mathcal{A}_s|\}$  // Pruning 4
9      $\mathcal{A}_c \leftarrow \{l \in \mathcal{A}_c \mid |\text{VERT}(l) \cap \mathcal{V}_r| \geq \max\left(\kappa, \lceil \frac{3 \log(\gamma)}{\log(3)} \rceil\right)\}$ 
10    // Pruning 4
11    while  $\mathcal{A}_c \neq \emptyset$  do
12      Pick and remove an element  $y$  from  $\mathcal{A}_c$ 
13       $\mathcal{V}_r' \leftarrow \mathcal{V}_r \cap \text{VERT}(y)$ 
14       $\mathcal{A}_s' \leftarrow \mathcal{A}_s \cup \{y\}$ 
15      HCSs  $\leftarrow \text{HCSs} \cup \text{ExploreSubgraphs2}(\mathcal{G}_e[\mathcal{V}_r'], \mathcal{A}_s', \mathcal{V}_r', \mathcal{A}_c)$ 
16 return R
  
```

It should be pointed out that a possible extension of Pruning 4 is to propagate incrementally the reduction over \mathcal{V}_r and \mathcal{A}_c until no

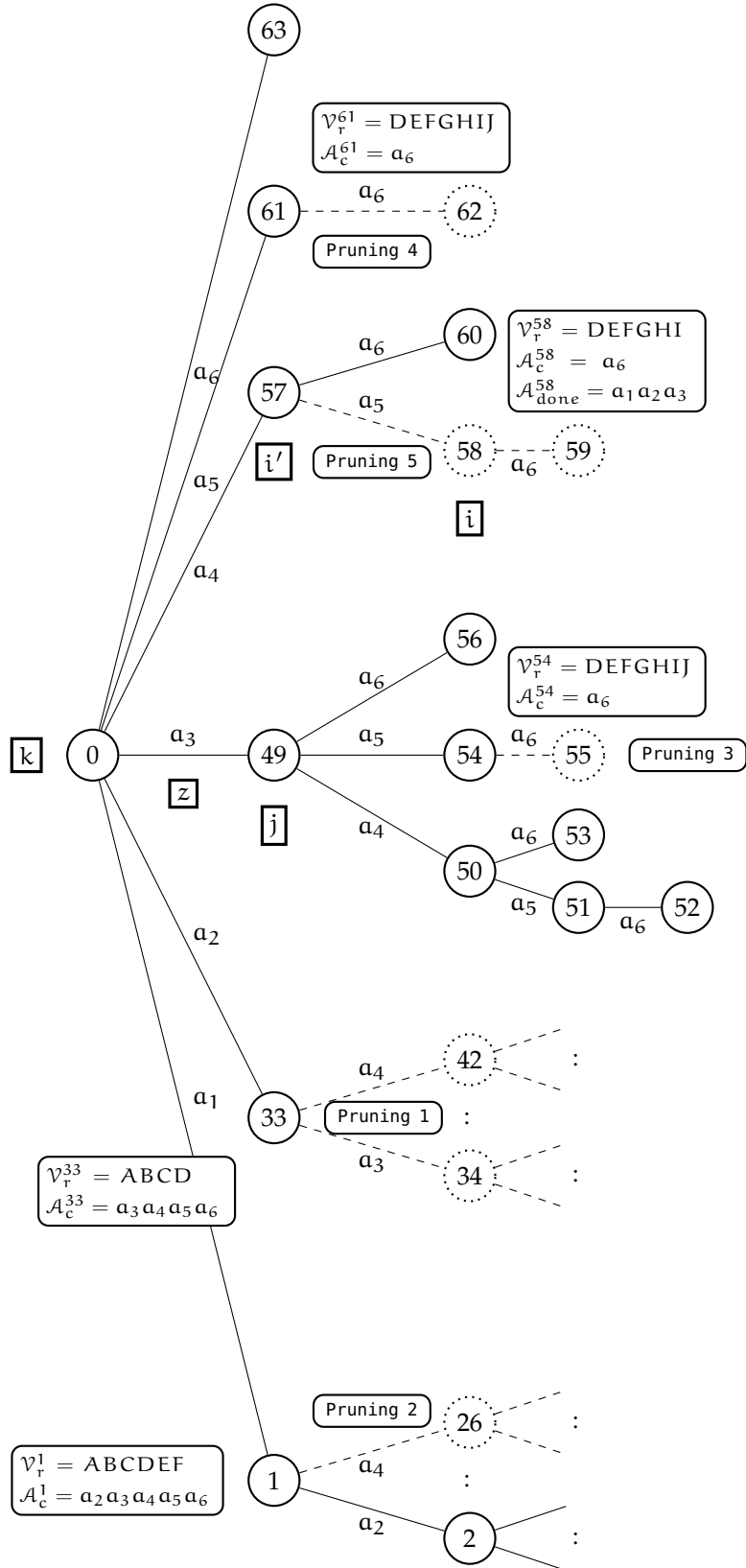


Figure 13.: Enumeration tree with pruning techniques 1, 2, 3, 4, and 5 corresponding to the attributed graph presented Figure 11 for constraints C_2^{homom} and $C_{2,3}^{clique}$.

more attribute or vertex can be removed. This extension, described as Algorithm 7, can replace lines 8 and 9 in Algorithm 6. However, when tested on real datasets and parameter settings corresponding to the experiments presented Section 4.3, the gain due to this extension is counterbalanced by its computing overhead (processing times with and without this extension were not significantly different). So, this incremental reduction has not been retained here.

Algorithm 7: Extension of Pruning 4

```

1 repeat
2    $\mathcal{V}_r^{\text{prev}} \leftarrow \mathcal{V}_r; \mathcal{A}_c^{\text{prev}} \leftarrow \mathcal{A}_c$ 
3    $\mathcal{V}_r \leftarrow \{v \in \mathcal{V}_r \mid |\text{ATB}(v) \cap \mathcal{A}_c| \geq \alpha - |\mathcal{A}_s|\}$ 
4    $\mathcal{A}_c \leftarrow \{a \in \mathcal{A}_c \mid |\text{VERT}(a) \cap \mathcal{V}_r| \geq \max\left(\kappa, \lceil \frac{3 \log(\gamma)}{\log(3)} \rceil\right)\}$ 
until  $\mathcal{V}_r^{\text{prev}} \neq \mathcal{V}_r$  or  $\mathcal{A}_c^{\text{prev}} \neq \mathcal{A}_c$  ;
```

The safety of the pruning techniques embedded in Algorithm 6 is established by the following lemma.

Lemma 4 *The property stated by Lemma 2 also holds for the enumeration trees obtained with Algorithm 6.*

Proof 5 *Pruning 1 and Pruning 4 remove subtrees that cannot lead to a HCS. Pruning 2 safely remove elements from \mathcal{A}_c^i (avoiding some duplicated graph during enumeration). Pruning 3 remove subtrees that cannot lead to a maximal HCS. So, for any node i in the enumeration tree, if there exists $L \subseteq \mathcal{A}_c^i$ and M a MHCS such that $M = \mathcal{C}_{\max}(\mathcal{G}_e^i[\text{VERT}(L)])$, then by Lemma 2 we still have the guaranty that M is obtained in the subtree rooted at node i .*

The fifth pruning technique (Pruning 5) avoids redundant enumeration of some MHCSs. It is presented in Algorithm 8 which is similar to Algorithm 6 apart from line 14 that is replaced by lines 13 to 15 in the new algorithm. This pruning uses a set $\mathcal{A}_{\text{done}}^i$ which contains for a node i the attributes used to build all immediate children of the ancestors of i apart from the attributes in the branch leading to i itself. For example, the set $\mathcal{A}_{\text{done}}^{54}$ corresponding to node 54 is $\{a_1, a_2, a_4\}$. The set $\mathcal{A}_{\text{done}}^i$ is updated line 15 of Algorithm 8. An example of enumeration tree with $\mathcal{A}_{\text{done}}$ values is given Figure 13. Given i a node corresponding to a call with input parameters $\mathcal{A}_{\text{done}}^i$ and \mathcal{V}_r^i such that there exists an attribute $z \in \mathcal{A}_{\text{done}}^i$ that is associated to all vertices of \mathcal{V}_r^i . Then, as stated by the following lemma, the MHCSs that could be obtained in the subtree rooted at i are redundant. Thus, in the call corresponding to the father of i , Pruning 5 line 13 of Algorithm 8 avoids the generation of i .

Lemma 5 (Redundancy of Algorithm 6) *Given i a node in the enumeration tree corresponding to Algorithm 6, and an attribute $z \in \mathcal{A}_{\text{done}}^i$ such that $\mathcal{V}_r^i \subseteq \text{VERT}(z)$. If M is a MHCS obtained in the subtree rooted at i , then M is also obtained in a subtree rooted at a node j , such that $j < i$ and j is not an ancestor of i .*

Proof 6 *Let M be a MHCS obtained in the subtree rooted at node i and let z be an attribute in $\mathcal{A}_{\text{done}}^i$ such that $\mathcal{V}_r^i \subseteq \text{VERT}(z)$. If M is obtained in the subtree, this implies that there exists a subgraph G of \mathcal{G}_e^i for which the*

algorithm computes M as $\mathcal{C}_{\max}(\mathcal{G}_e)$. Thus there exists $A \subseteq \mathcal{A}_c^i$ such that $M = \mathcal{C}_{\max}(\mathcal{G}_e^i[\text{VERT}(A)])$.

Let us now show that M is also obtained in a subtree rooted at a node j defined as follows. Let node k be the ancestor of i in which attribute z was picked, node j is the node corresponding to the call made from k for this attribute z . Finally, let node i' be the child of k in the branch leading to i (notice that i' can be i itself if i is a child of k). Figure 13 gives an example of such nodes with $i = 58$, $k = 0$, $i' = 57$, $j = 49$, and $z = a_3$.

Let $V = \{v \in \mathcal{V}_r^k \mid |\text{ATB}(v) \cap \mathcal{A}_c^k| \geq \alpha - |\mathcal{A}_s^k|\}$ be the reduced set of vertices computed line 8 during the call corresponding to node k . The call corresponding to j is made with $\mathcal{V}_r^j = V \cap \text{VERT}(z)$. As $\mathcal{V}_r^i \subseteq V$ (V is used in node k to generate i' ancestor of i) and $\mathcal{V}_r^i \subseteq \text{VERT}(z)$ (by hypothesis), then $\mathcal{V}_r^i \subseteq \mathcal{V}_r^j$. Thus \mathcal{G}_e^i is an induced subgraph of \mathcal{G}_e^j , and then $M = \mathcal{C}_{\max}(\mathcal{G}_e^j[\text{VERT}(A)])$.

Since i' has the same father as j and $j < i'$, we have $\mathcal{A}_c^{i'} \subset \mathcal{A}_c^j$ and thus $\mathcal{A}_c^i \subset \mathcal{A}_c^j$ (i' is an ancestor of i). So, $A \subseteq \mathcal{A}_c^j$.

Whence, there exists $A \subseteq \mathcal{A}_c^j$ such that $M = \mathcal{C}_{\max}(\mathcal{G}_e^j[\text{VERT}(A)])$, and by Lemma 4, we know that M is obtained in the subtree rooted at j .

The completeness of Algorithm 8 is stated by the following lemma.

Lemma 6 *In the enumeration tree explored by Algorithm 8, if a subtree T rooted at a node i (including i itself) is pruned using Pruning 5 and a MHCS M would be obtained in T then M is obtained in a subtree rooted at node j , with $j < i$ and j is not an ancestor of i .*

Proof 7 *Proof is immediate by strong induction. The property holds for $n = 0$, i.e. the root of the whole enumeration tree, since the root cannot be pruned by Pruning 5. Let n be any node in the enumeration tree. Suppose that for all node $k < n$ Lemma 6 holds. Let us show that the property also holds for n . Let T be a subtree rooted at n and pruned using Pruning 5. Let M be a MHCS that would be obtained in T . Then from Lemma 5 there is a subtree rooted at $n' < n$ in which M is obtained by Algorithm 6. If this branch has not been pruned by Pruning 5, then we obtained M with Algorithm 8. If this branch has been pruned by Pruning 5, then by induction hypothesis, there exists a subtree rooted at $n'' < n'$ in which M is obtained with Algorithm 8.*

Notice that Pruning 5 can also avoid the enumeration of non-maximal HCSs, but in this case there is no need to prove that they have already been obtained, since we are not interested in such HCSs.

Example 21 (Enumeration tree with Pruning 5) *In the enumeration tree using Pruning 5 presented Figure 13, consider node 58. The set $\mathcal{A}_{\text{done}}^{58}$ contains all the attributes used to build all immediate children of the ancestor of node 58 apart from the attributes in the branch leading to node 58, i.e., $\mathcal{A}_{\text{done}}^{58} = \{a_1, a_2, a_3\}$. Considering attribute a_3 in $\mathcal{A}_{\text{done}}^{58}$, we have $\text{VERT}(a_3) = \{A, D, E, F, G, H, I, J\}$ and $\mathcal{V}_r^{58} = \{D, E, F, G, H, I\}$. As all vertices in \mathcal{V}_r^{58} are also in $\text{VERT}(a_3)$, the pruning criterion for Pruning 5 is satisfied line 13 during the call corresponding to node 57, and thus node 58 is pruned.*

The following theorem states the correctness of the final extraction algorithm given as Algorithm 9. It is a direct consequence of Lemma 6.

Theorem 3 *Algorithm 9 returns all MHCSs and only MHCSs.*

Algorithm 8: ExploreSubgraphs

Input: $\mathcal{G}_e, \mathcal{A}_s, \mathcal{V}_r, \mathcal{A}_c, \mathcal{A}_{\text{done}}$

```

1 HCSs  $\leftarrow \emptyset$ 
2 if  $\{C \in \mathcal{C}_{\text{max}}(\mathcal{G}_e) \mid |C| \geq \kappa\} \geq \gamma$  then // Pruning 1
3    $S \leftarrow \{l \in \mathcal{A}_c \mid \mathcal{V}_r \subseteq \text{VERT}(l)\}$  // Pruning 2
4    $\mathcal{A}_c \leftarrow \mathcal{A}_c \setminus S; \mathcal{A}_s \leftarrow \mathcal{A}_s \cup S$  // Pruning 2
5   if  $|\mathcal{A}_s| \geq \alpha$  then
6      $\text{HCSs} \leftarrow \{\mathcal{C}_{\text{max}}(\mathcal{G}_e)\}$ 
7   else // Pruning 3
8      $\mathcal{V}_r \leftarrow \{v \in \mathcal{V}_r \mid |\text{ATB}(v) \cap \mathcal{A}_c| \geq \alpha - |\mathcal{A}_s|\}$  // Pruning 4
9      $\mathcal{A}_c \leftarrow \{l \in \mathcal{A}_c \mid |\text{VERT}(l) \cap \mathcal{V}_r| \geq \max(\kappa, \lceil \frac{3 \log(\gamma)}{\log(3)} \rceil)\}$ 
10    // Pruning 4
11    while  $\mathcal{A}_c \neq \emptyset$  do
12      Pick and remove an element  $y$  from  $\mathcal{A}_c$ 
13       $\mathcal{V}'_r \leftarrow \mathcal{V}_r \cap \text{VERT}(y)$ 
14       $\mathcal{A}'_s \leftarrow \mathcal{A}_s \cup \{y\}$ 
15      if  $\forall z \in \mathcal{A}_{\text{done}}, \mathcal{V}'_r \not\subseteq \text{VERT}(z)$  then // Pruning 5
16         $\text{HCSs} \leftarrow \text{HCSs} \cup \text{ExploreSubgraphs}(\mathcal{G}_e[\mathcal{V}'_r], \mathcal{A}'_s, \mathcal{V}'_r,$ 
17           $\mathcal{A}_c, \mathcal{A}_{\text{done}})$ 
18         $\mathcal{A}_{\text{done}} \leftarrow \mathcal{A}_{\text{done}} \cup \{y\}$ 
19 return HCSs

```

Algorithm 9: ExtractMHCS

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \text{ATB})$

```

1 HCSs  $\leftarrow \text{ExploreSubgraphs}(\mathcal{G}, \emptyset, \mathcal{V}, \mathcal{A}, \emptyset)$ 
2 output RemoveNonMaximalHCSs(HCSs)

```

4.2.3 Implementation

The RemoveNonMaximalHCSs post-processing used to remove from a collection of HCSs non maximal ones has been implemented using the two following optimizations. First, it is not necessary to check maximality over the whole collection, since the HCSs whose vertices share exactly α labels are necessarily maximal and do not need to be tested. The second optimization is based on the following property. To compare two HCSs it is not required to test the pairwise inclusion of all cliques in the two HCSs, but it is sufficient to test the inclusion of the unions of these cliques: given M_1 and M_2 , two HCSs, $M_1 \prec M_2$ if $\bigcup_{C_1 \in M_1} C_1 \subset \bigcup_{C_2 \in M_2} C_2$. In our experiments, the runtime of RemoveNonMaximalHCSs was negligible with respect to the extraction time. Post-processing runtimes are given in Section 4.3.

The algorithm used to compute maximal cliques is CLIQUES [84] presented page 21 as Algorithm 2. Two improvements were made to use CLIQUES in our main extraction algorithm. The first one is used on line 2 in Algorithm 8 to avoid the computation of all maximal cliques. The algorithm stop once a set of cliques satisfying $\mathcal{C}_{\gamma, \kappa}^{\text{clique}} \wedge \mathcal{C}^{\text{sep}}$ has been found. The second optimization consists in verifying that there is at least $\max(\kappa, \lceil \frac{3 \log(\gamma)}{\log(3)} \rceil)$ vertices connected to at least $\kappa - 1$ other vertices in the enumerated graph, otherwise it is not possible to satisfy $\mathcal{C}_{\gamma, \kappa}^{\text{clique}} \wedge \mathcal{C}^{\text{sep}}$.

4.3 EXPERIMENTS

In this section we report the results of our experiments. They were performed using a network of collaborations of researchers associated to the conference and journals where they have published. Another set of experiments in the context of molecular biology is presented in Section 6. Experiments were also realized on synthetic datasets to measure the impact of graph structure over time performance, and are described in Section 4.3.4.

All experiments were performed on a PC running GNU/Linux with a 3 GHz Core 2 Duo CPU and 8 GB of main memory installed (no more than 800 MB used in the experiments as shown in Section 4.3.3). The algorithm has been implemented using Scala 2.9¹. This algorithm has been embedded in a tool, described in Appendix B, that allows to browse and visualise the collections of MHCs.

This section is organized as follows. First we present the datasets, then we illustrate the interest of MHCS in real data. The next section presents quantitative results with respect to runtime, memory usage and number of patterns extracted. The impact of graph structure over performance is studied later using a synthetic dataset. Finally we present the runtime improvement for each pruning technique with respect to a baseline algorithm.

4.3.1 DBLP: a scientific collaboration network dataset

Here we give a short description of the scientific collaboration network datasets. We use the public DBLP database². This database contains rather exhaustive bibliographic information on most computer science conferences and journals. It has been extensively used as an experimental dataset by many researchers. From this database, we built three attributed graphs, DBLP₁, DBLP₂, and DBLP₃. Several graph characteristics are presented on Table 1 and an exhaustive description is given in Appendix A. The vertices in an attributed graph correspond to authors, an edge representing a coauthor relationship. The attributes are the conferences and journals where the authors have published. All data from DBLP up to august 2011 was used to build the datasets.

DBLP₁ and DBLP₂ are used to assess the performances of the algorithm. Consequently, we wanted large datasets even if the extracted patterns might not be very meaningful. On the other hand, in DBLP₃ we kept only what can be considered as strong cooperation between researchers and strong engagement in a specific research field. DBLP₁ contains all coauthor relationships, and an author is associated to all journals and conferences where she/he has published at least once (all editions of a conference are aggregated under the same conference name). In DBLP₂ (resp. DBLP₃) we have an edge between two authors only if they have coauthored at least two (resp. three) articles, and an author is associated to journals and conferences where she/he has published at least two (resp. three) times. In DBLP₂ and DBLP₃, authors with empty attribute list are removed (authors that have never published twice/three times in the same journal or conference).

1. Scala is a language running over a Java virtual machine.

2. <http://dblp.uni-trier.de/>

	DBLP ₁	DBLP ₂	DBLP ₃
# Vertices	997,050	266,125	127,386
# Attributes	5,963	5,309	3,980
# Edges	3,427,683	650,205	234,896
Avg. degree	6.88	4.89	3.69
Maximum degree	1014	240	149
Avg. attributes/vertex	3.06	2.44	2.15
Maximum attributes	302	112	68

Table 1.: Measures describing datasets DBLP₁, DBLP₂, and DBLP₃.

4.3.2 Interpretation of MHCSs from DBLP

The interest of our approach is illustrated with two patterns extracted from DBLP₃. On this dataset, we first search for MHCSs formed by relatively large communities so, κ , the minimal size of the core cliques, is set to 5. We want at least two groups of core cliques (*i.e.*, $\gamma = 2$) and we do not require them to be very homogeneous so we set α to 2. With this parameter setting, we found 498 MHCSs. Among them, we focus on the patterns related to the conference IEEE International Conference on Computer Communications (INFOCOM). It is a major conference on the topic of computer communications. Five MHCSs among the extracted collection are related to this conference and we present in Figure 14 the one having the smallest number of vertices.

This pattern contains 18 maximal cliques, two containing at least five vertices (*i.e.*, the core cliques):

- {Yuan He, Mo Li, Xiang-Yang Li, Yurhao Liu, Zheng Yang}, bottom left of Figure 14
- {Tarek F. Abdelzaher, Qing Cao, Lin Gu, Tian He, Liqian Luo, John A. Stankovic, Gang Zhou}, top of Figure 14

Using ArnetMiner and the web pages of the authors, we consider the affiliations of the authors. The first core clique is formed by authors who have all been working in Virginia, while the second core clique is formed by four out of five authors affiliated to the university of Tsinghua in China (the fifth author is also affiliated to a university in China). A third group of vertices at the bottom left of Figure 14 is structured as several 3-cliques with large overlaps. It corresponds to seven authors: Deborah Estrin, Ramesh Govindan, John S. Heidemann, Ahmed Helmy, Polly Huang, Bhaskar Krishnamachari, and Scott Shenker. All these authors are currently affiliated to universities in the west coast of the United States of America, four being more particularly located in the university of Southern California.

We also consider MHCSs being more homogeneous and containing more core cliques while relaxing the constraint on the size of the core cliques. To perform the extraction, we looked for patterns with at least 5 cliques of 3 vertices and 6 shared attributes ($\alpha = 6$, $\kappa = 3$, and $\gamma = 5$). Using this parameter setting, 718 patterns were extracted. Among them, 8 patterns are related to the data mining conference IEEE International Conference on Data Mining (ICDM). We present in Figure 15 one of the extracted MHCSs having the smallest number of vertices.

ArnetMiner
(<http://arnetminer.org/>)
is a web site used to
index and search
academic networks.

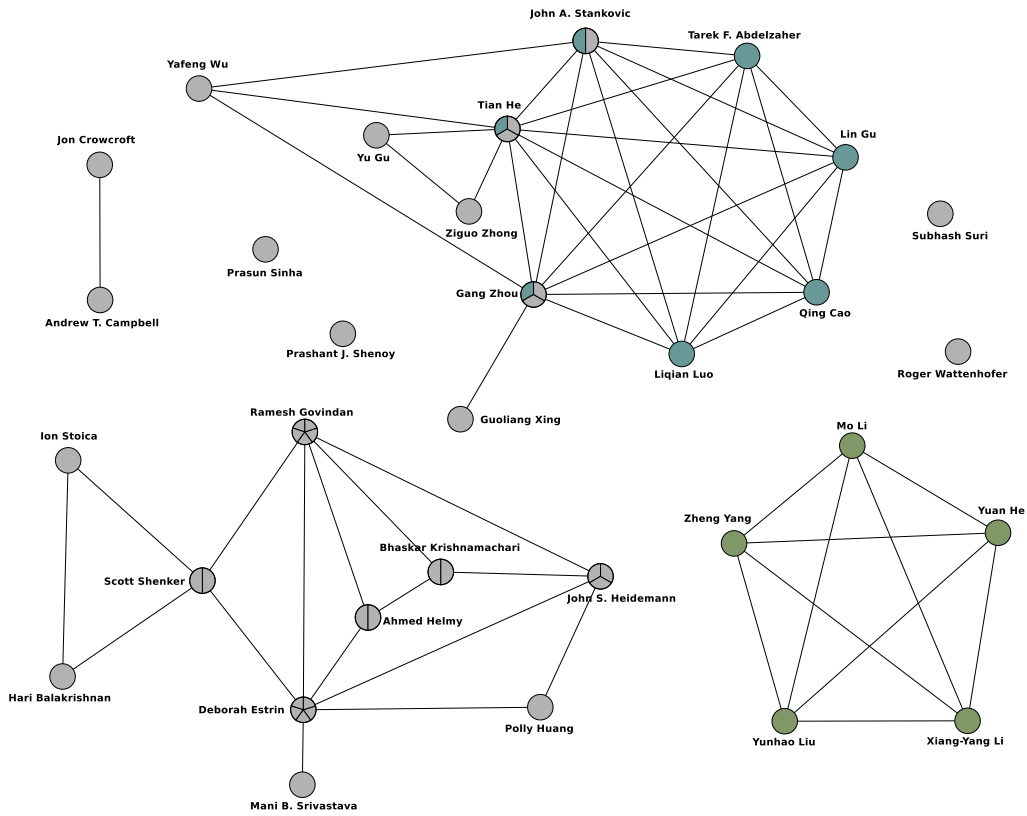


Figure 14.: A pattern related to conferences INFOCOM and SenSys. Each colour denotes a clique of at least κ vertices. A vertex in several colours is contained in multiple cliques. Vertices in light grey are not contained in a clique of at least κ vertices.

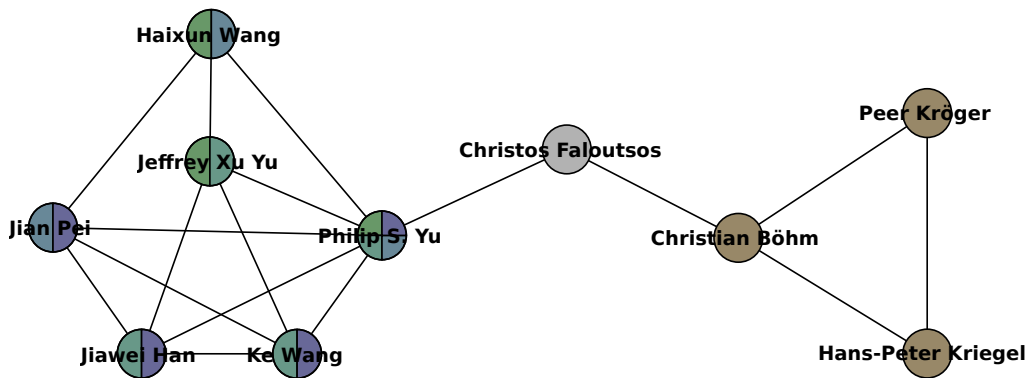


Figure 15.: A pattern related to conferences ACM SIGMOD, CIKM, EDBT, ICDE, ICDM, and SDM. Each colour denotes a clique of at least κ vertices. A vertex in several colours is contained in multiple cliques. Vertices in light grey are not contained in a clique of at least κ vertices.

	DBLP ₁	DBLP ₂	DBLP ₃
Mean	1,207 Mb	421 Mb	198 Mb
Max	1,886 Mb	735 Mb	416 Mb
Standard deviation	175 Mb	84 Mb	46 Mb

Table 2.: Memory consumption over all experiments reported Figure 16.

This pattern contains seven maximal cliques and, among them, five contains at least three vertices (the core cliques):

- {Jiawei Han, Jian Pei, Ke Wang, Philip S. Yu}
- {Haixun Wang, Jeffrey Xu Yu, Philip S. Yu}
- {Jiawei Han, Ke Wang, Jeffrey Xu Yu, Philip S. Yu}
- {Jian Pei, Haixun Wang, Philip S. Yu}
- {Christian Böhm, Hans-Peter Kriegel, Peer Kröger}

Two groups are forming this pattern. One is formed by Christian Böhm, Hans-Peter Kriegel and Peer Kröger, all these authors working in the same university located in Germany. The second group is formed by people located in North America (working in the same universities at some time). These two groups are connected by Christos Faloutsos, who is not part of any κ -clique in this pattern but still associated to the same conferences. This vertex has a betweenness centrality measure of 4.5 (the betweenness centrality measure is presented page 20 in the state of the art).

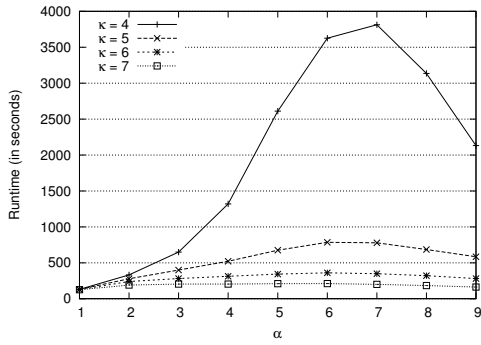
The main interest of such MHCSs is to exhibit a local structure of groups sharing similar interests. Knowing such structures can be useful, for instance, to help reviewer selection for projects, or to set up new scientific collaborations (a task also considered in [7], using different approaches).

4.3.3 Performance study on DBLP datasets.

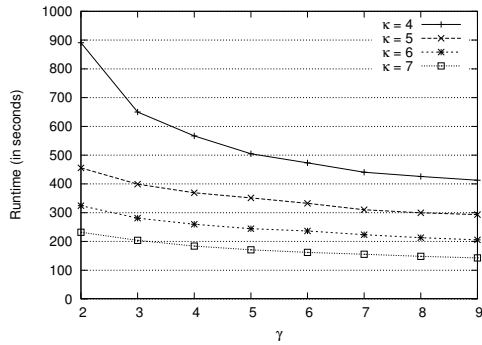
The runtime, number of extracted patterns, and maximal memory consumption are reported for extractions done on DBLP₁, DBLP₂, and DBLP₃ with different parameter settings.

Concerning time performances, Figure 16 shows that the extractions can be made in less than one minute on DBLP₂ and DBLP₃ even when constraints are weakly selective. On DBLP₁, the runtime is presented only for $\kappa \geq 4$. The worst case is obtained for $\alpha = 7$, $\kappa = 4$, and $\gamma = 3$ and requires about one hour. Presented runtime take into account the post-processing needed to remove non-maximal MHCSs. This post-processing requires less than one second for all reported experiments.

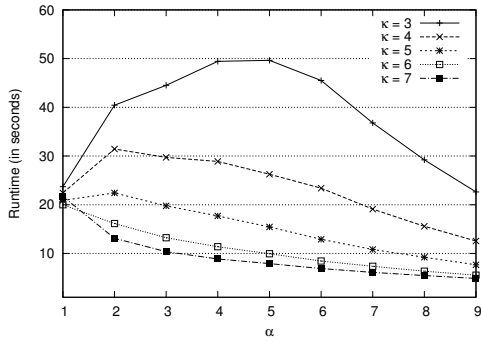
When α (*i.e.*, the minimum number of shared attributes) increases, runtime increases until it reach a maximum, then start to decrease. This behaviour is due to the fact that at first there are more combinations of attributes of size $n + 1$ compares to the combinations of size n , thus maximal cliques have to be compute on more subgraphs and runtime increases with α . In a second phase, the runtime decreases when α increases since large set of attributes are less likely to be shared by enough vertices to satisfy $C_{\gamma, \kappa}^{\text{clique}}$, and thus are more likely to be pruned during the enumeration. Moreover for κ, γ fixed and large values of α , the runtime tends to stabilize. One can also notice that for



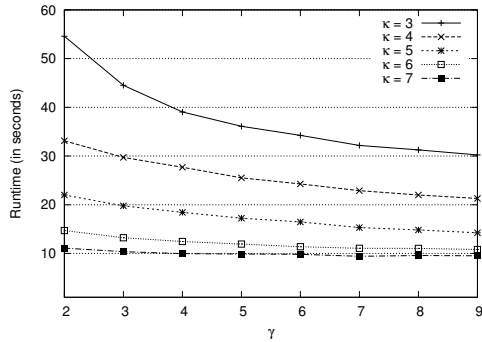
(a) Runtimes on DBLP₁ with $\gamma = 3$



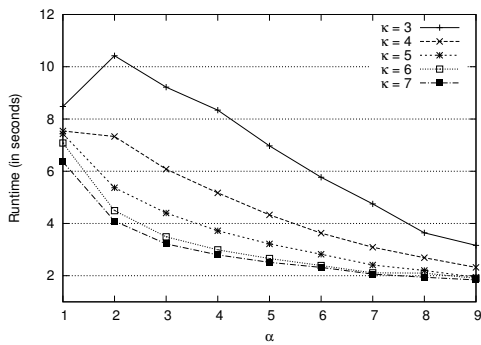
(b) Runtimes on DBLP₁ with $\alpha = 3$



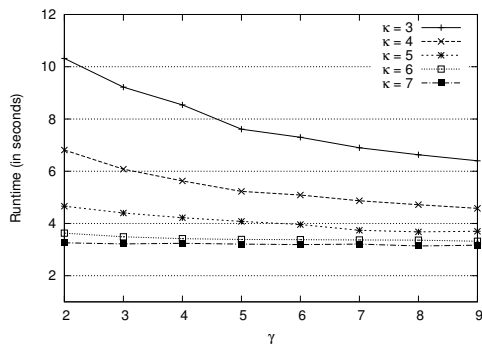
(c) Runtimes on DBLP₂ with $\gamma = 3$



(d) Runtimes on DBLP₂ with $\alpha = 3$



(e) Runtimes on DBLP₃ with $\gamma = 3$



(f) Runtimes on DBLP₃ with $\alpha = 3$

Figure 16.: Runtime for different sets of parameters on DBLP₁, DBLP₂, and DBLP₃.

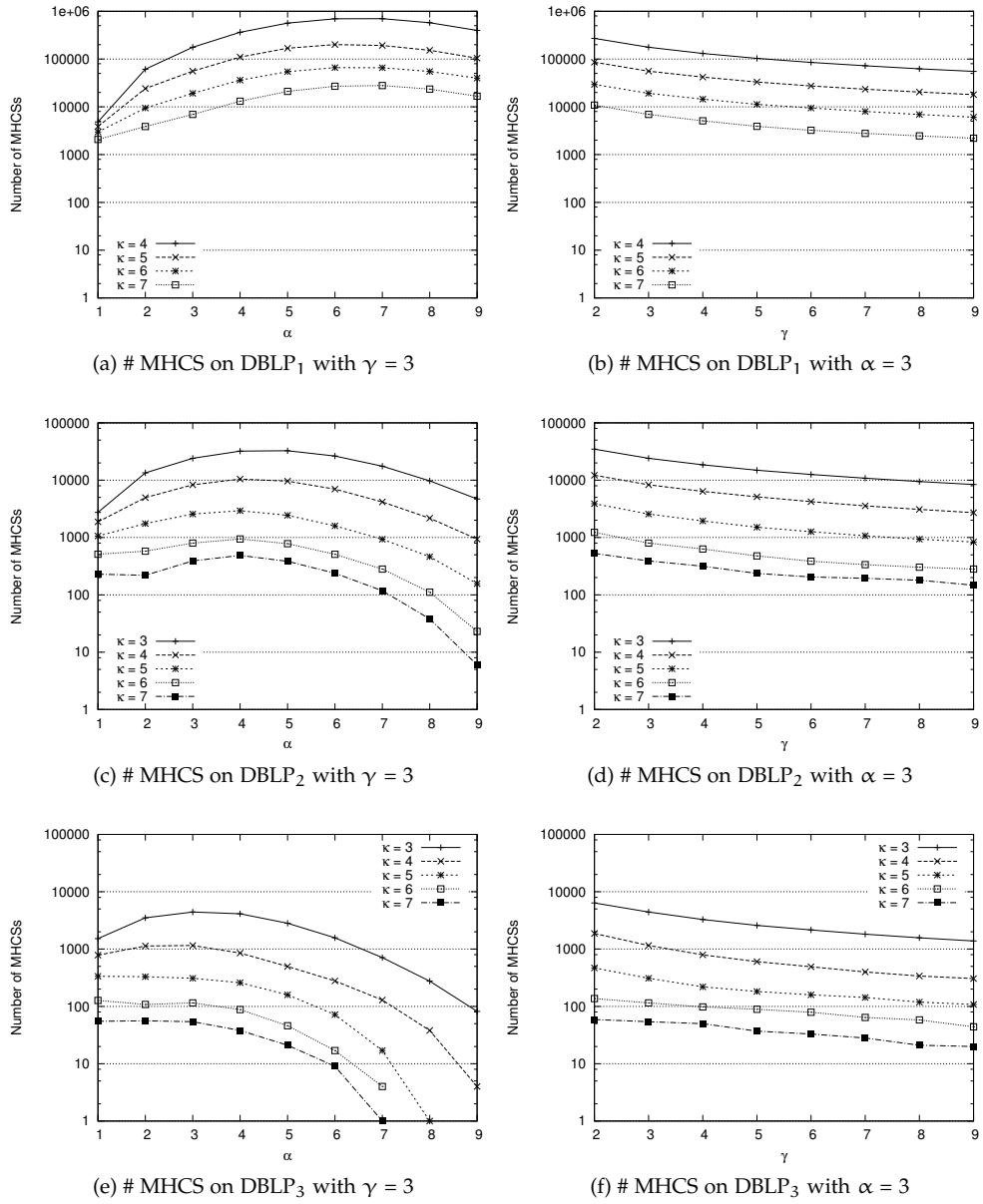


Figure 17.: Number of MHCS for different sets of parameters on DBLP₁, DBLP₂, and DBLP₃.

$\alpha = 1$, the values of κ seems to have a smaller impact on runtime. This is because when $\alpha = 1$ the enumeration stops on the second level of the enumeration tree and the pruning based on γ or κ are not used.

Regarding the number of output patterns, Figure 17 shows that it shrinks fast when parameters κ , α , or γ increase. For values of $\kappa \leq 5$ and $\alpha > 1$, increasing κ by two reduces the size of the collection of patterns by one order of magnitude.

Regarding main memory usage, only maximal memory consumption during each extraction is considered. Table 2 summarizes the results for all experiments reported in this section.

4.3.4 Evaluation on synthetic datasets

In this section we describe an experimental evaluation of the algorithm using synthetic datasets. The goal of these experiments is to study runtime evolution with regards to different graph structures and user parameters. The generation of synthetic data which model accurately graph structure is an active research area. Several models has been proposed to mimic graph structures, however as far as we know, only two approaches allows the generation of attributed graphs [50, 58]. Both models use an existing attributed graph which determines the structure of the generated data. As we wanted to set the parameters given on Table 3 for each datasets, these models did not fit our needs.

The model used to generate synthetic data is simple. An attributed graph is first generated with four parameters:

- #vert: number of vertices
- #attr: number of different attributes
- avgDeg: average vertex degree
- avgAtt: average number of attributes having value True per vertex

For the generation of the vertices and edges, we used the well studied Erdős-Rényi random graph model [22]. This model requires two parameters, the first is simply the number of vertices #vert, and the second one is the probability p for each pair of vertices to be connected by an edge. We set p to $\#edges/\#edgesMax$ where $\#edges$ was the expected number of edges and was equal to $\#vert \times avgDeg/2$, and $\#edgesMax$ was the maximum number of possible edges, *i.e.*, $\#edgesMax = \#vert \times (\#vert - 1)/2$.

Then, the attributes were associated randomly to the vertices as follows: for each vertex v and each attribute x , x was associated to v with the probability $avgAtt/\#attr$.

For the generation, we took a reference parameter setting that was close to the characteristics of the BioData₄₀₀ dataset used in the experiments presented in Section 6 in order to start from a real setting. However, we did not intend to mimic the BioData₄₀₀ structure (performances on this kind of structures are presented in Section 6). The values retained for the reference parameters were #vert = 15,000; #attr = 500; avgDeg = 20; and avgAtt = 10.

Due to the random generation process, it is unlikely that such synthetic datasets contain some MHCS, except trivial ones. Thus a number #hcs of HCSs were injected in the dataset. These HCSs were generated randomly according to three parameters k , g and s , as follows. Each of these HCSs contained $k \times g$ vertices, structured in the form of g cliques of size k , and with s attributes shared by all vertices. Note that the injected HCSs might be non-maximal in the resulting dataset if the orig-

	$S_{\#vert}$	$S_{\#attr}$	S_{avgDeg}	S_{avgAtt}	$S_{\#hcs}$
#vert	5,000-25,000	15,000	15,000	15,000	15,000
#attr	500	200-800	500	500	500
avgDeg	20	20	10-30	20	20
avgAtt	10	10	10	5-25	10
#hcs	300	300	300	300	100-500

Table 3.: Parameters used to generate the synthetic datasets.

inal graph contains vertices sharing the same set of attributes. However, except for trivial cases (low values of #attr and s), this is not likely to occur, and then these injected HCSs are expected to be MHCSs. The number of injected patterns #hcs was used as an additional generation parameter, and we used #hcs = 300 as a reference value.

We generated several datasets, by changing in turn each parameter (#vert, #attr, avgDeg, avgAtt and #hcs), within a range encompassing its reference value. Let var be one of the parameter, we denote S_{var} the collection of datasets obtained by varying this parameter var. For each collection, the constant parameter values and the range of the parameter that was changed, are given in Table 3. In each dataset, we did not only perform a single pattern injection, but we built four derived datasets by adding four different sets of random HCSs using the four following parameters settings: (1) $s = 2, k = 6, g = 2$, (2) $s = 2, k = 6, g = 4$, (3) $s = 4, k = 6, g = 2$ and (4) $s = 4, k = 6, g = 4$.

Then, the MHCSs were extracted on each of these derived datasets using parameters $\alpha = s, \gamma = g$ and $\kappa = k$, so as to retrieve the patterns that have been injected. The runtimes and the precise values used within each range for #vert, #attr, avgDeg, avgAtt and #hcs can be found Figure 18. Each point of the graphs of Figure 18 corresponds to the average of the runtime over ten different random generations of the data. The runtime remained stable with a maximal standard deviation of 0.26 for each point. The curves show that the extraction were tractable in practice on this graph model, for a wide range of parameter values.

Concerning the number of extracted patterns, the number of MHCSs obtained was always equal to the number of injected HCSs, except for a few experiments. In the worst case, we had a difference of 6 patterns. For each apparently missing pattern, we observed that this difference came from two injected HCSs sharing the same attributes and forming a single MHCS, and thus only one pattern was obtained instead of two.

4.3.5 Comparison of the prunings to baseline algorithms

In this section we compared the effects of each pruning techniques (pruning 1 to 5 introduced Section 4.2) on runtime. In order to perform this comparison, we proposed five versions of the extraction algorithm based on a baseline algorithm. This baseline algorithm is similar to the generate and test version presented as Algorithm 4 except for two simple additional verifications. First, a test ensures that \mathcal{V}_r contains at least κ vertices (less than κ vertices cannot form a pattern satisfying $C_{\gamma, \kappa}^{clique}$). A second test checks that the number of shared attributes or

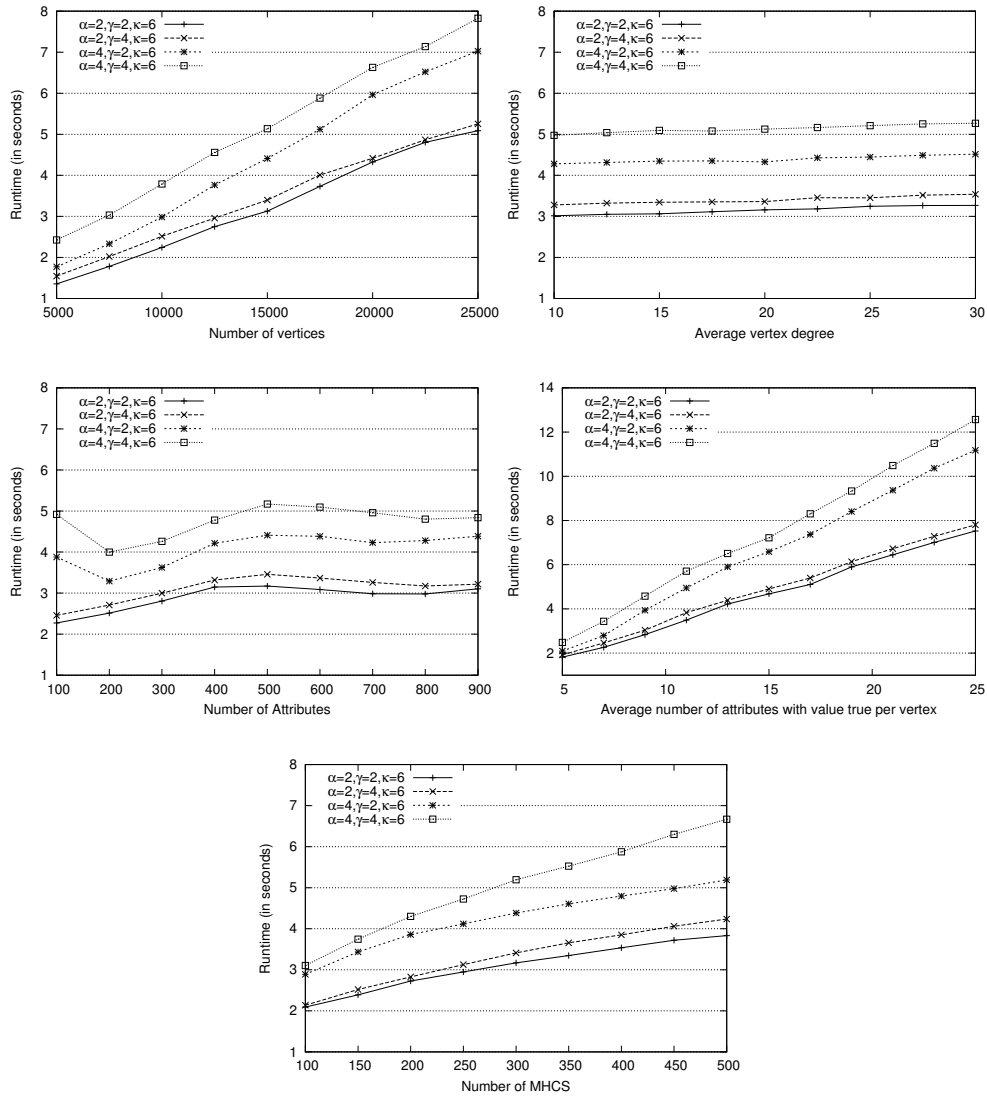


Figure 18.: Runtime for the collections of datasets $S_{\#vert}$, S_{avgDeg} , $S_{\#attr}$, S_{avgAtt} and $S_{\#hcs}$.

candidate attributes (*i.e.*, $|\mathcal{A}_c \cup \mathcal{A}_s|$) is at least α , otherwise the homogeneity constraint cannot be satisfied (this baseline algorithm is given as Algorithm 10).

Algorithm 10: Baseline

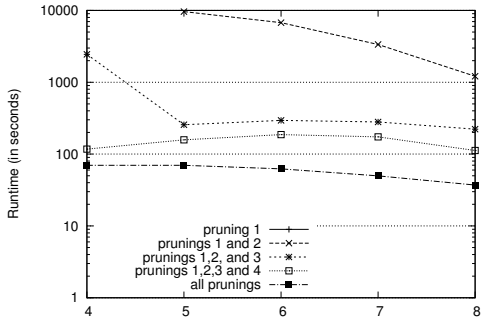
Input: $\mathcal{G}_e, \mathcal{A}_s, \mathcal{V}_r, \mathcal{A}_c$

- 1 $\text{HCSs} \leftarrow \emptyset$
- 2 **if** $|\mathcal{A}_s| \geq \alpha$ **and** $|\{C \in \mathcal{C}_{\max}(\mathcal{G}_e) \mid |C| \geq \kappa\}| \geq \gamma$ **then**
- 3 $\text{HCSs} \leftarrow \{\mathcal{C}_{\max}(\mathcal{G}_e)\}$
- 4 **while** $\mathcal{A}_c \neq \emptyset$ **do**
- 5 Pick and remove an element y from \mathcal{A}_c
- 6 $\mathcal{V}'_r \leftarrow \mathcal{V}_r \cap \text{VERT}(y)$
- 7 $\mathcal{A}'_s \leftarrow \mathcal{A}_s \cup \{y\}$
- 8 **if** $|\mathcal{V}'_r| \geq \kappa$ **and** $|\mathcal{A}_c \cup \mathcal{A}_s| \geq \alpha$ **then**
- 9 $\text{HCSs} \leftarrow \text{HCSs} \cup \text{Baseline}(\mathcal{G}_e[\mathcal{V}'_r], \mathcal{A}'_s, \mathcal{V}'_r, \mathcal{A}_c)$
- 10 **return** HCSs

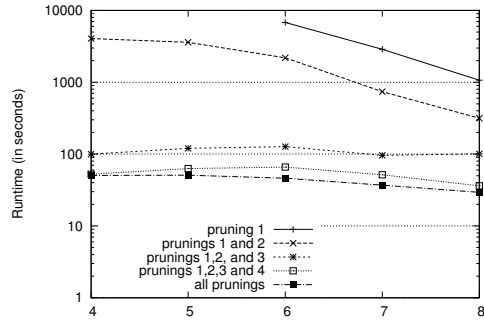
We incrementally added the pruning techniques to the baseline algorithm, starting from Pruning 1 to Pruning 5. The version incorporating all prunings from 1 to 5 is then the same as the one presented as Algorithm 8 page 53.

The experiments were only run over DBLP₂ and DBLP₃ since extraction runtimes using only Pruning 1 were prohibitive on DBLP₁. The results are presented Figure 19. We observe that adding step by step the prunings 1 to 4 improves the runtime in most cases. Considering Pruning 5, one can notice that when the runtime of prunings 1 + 2 + 3 + 4 is low, adding Pruning 5 can slightly increase the runtime, but when the runtime of prunings 1 + 2 + 3 + 4 is larger, then adding Pruning 5 can reduce it substantially (see Figure 19a).

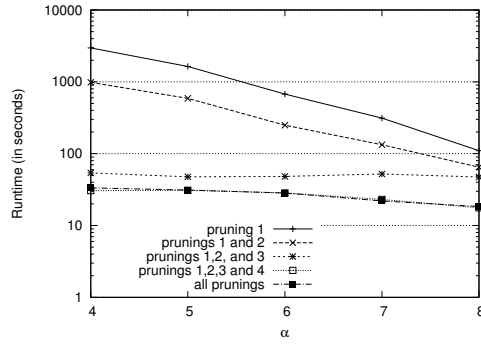
In the next section, we will now introduce a second family of patterns related to the MHCS but being less restrictive on the connectivity of the different groups of vertices in the patterns.



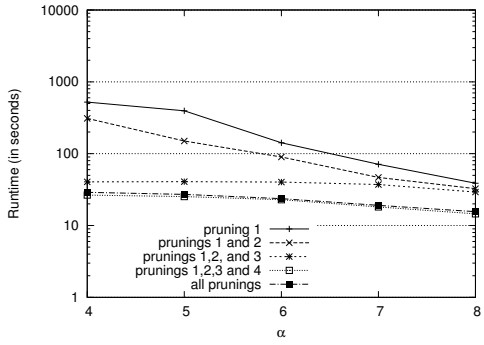
(a) DBLP₂ with $\gamma = 2$ and $\kappa = 3$.



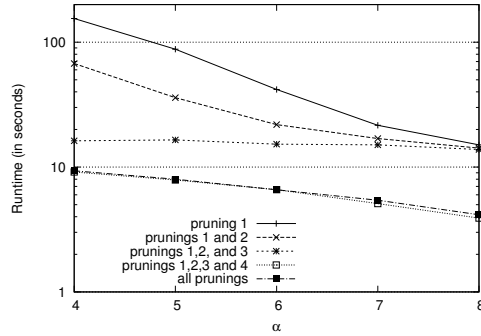
(b) DBLP₂ with $\gamma = 3$ and $\kappa = 3$.



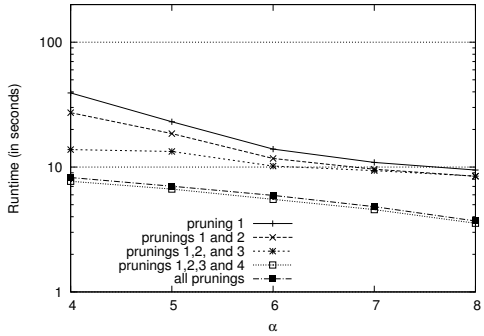
(c) DBLP₂ with $\gamma = 2$ and $\kappa = 4$.



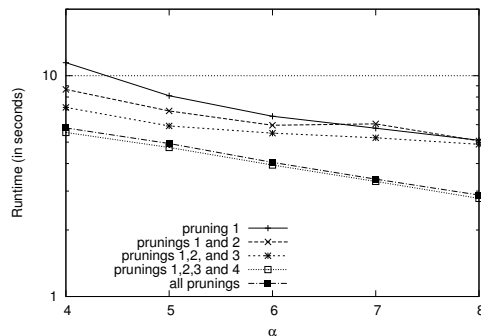
(d) DBLP₂ with $\gamma = 3$ and $\kappa = 4$.



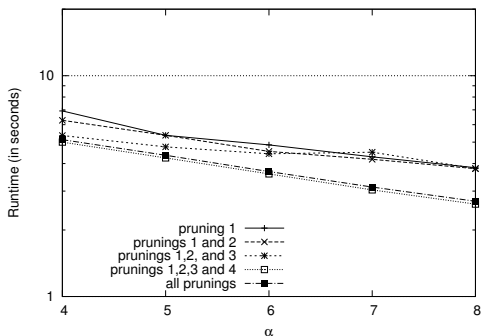
(e) DBLP₃ with $\gamma = 2$ and $\kappa = 3$.



(f) DBLP₃ with $\gamma = 3$ and $\kappa = 3$.



(g) DBLP₃ with $\gamma = 2$ and $\kappa = 4$.



(h) DBLP₃ with $\gamma = 3$ and $\kappa = 4$.

Figure 19.: Runtime for different pruning techniques on the datasets DBLP₂ and DBLP₃. The scale is logarithmic for the runtime.

MINING COLLECTIONS OF K-CLIQUE PERCOLATED COMPONENTS

5.1 PATTERN DEFINITION

We now define a family of patterns similar to the MHCSs, but based on k -clique percolated components and being more tolerant on the connectivity.

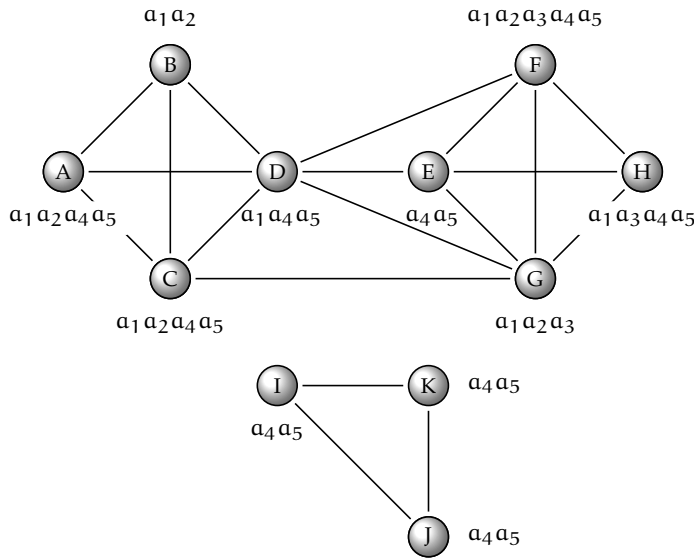


Figure 20.: A Boolean attributed graph illustrating the CoHoP patterns. Vertices are identified by capital letters and attributes are identified by a_i , $i \in \{1, \dots, 5\}$. Only attributes having value True are displayed.

As mentioned in the state of the art page 24, a k -clique percolated component (termed also k -clique-community), k -PC for short, introduced in [67] is a relaxed version of the cliques. A k -PC is the union of all k -cliques that can be reached from each other through a series of adjacent k -cliques. Compared to other fault-tolerant clique definitions, a particularity of the k -PCs is to enforce the fact that each vertex can be reached from any other vertex through highly connected subset of vertices [67]. In the context of social networks, it represents a community of individuals where each person, even if not directly connected to another member, can easily find a way to communicate with him/her.

Example 22 Let us look for 4-PCs in the attributed graph presented Figure 20. It contains three 4-cliques (i.e., $k = 4$): ABCD, DEFG, and EFGH. As DEFG and EFGH share three vertices, these two cliques form a 4-PC. The clique ABCD does not share enough vertices with the other cliques so the collection of 4-PCs is {ABCD, DEFGH}. If we look for 3-PCs, all single 3-clique except IJK can be merged. So, the collection of 3-PCs is {ABCDEFGH, IJK}.

We now define a new family of patterns similar to the MHCSs, but based on k -PCs. Let k , α , and γ be three strictly positive integers, we

define a pattern called a Collection of Homogeneous k-PCs (CoHoP) as a pattern satisfying three conditions:

1. the vertices are homogeneous, in the sense that they share at least α attributes;
2. the collection contains at least γ k-PCs;
3. all k-PCs sharing the same set of attributes are in the collection.

These patterns are defined more precisely as follows.

Definition 19 (Collection of Homogeneous k-PCs (CoHoP)) *Let k , α , and γ be three strictly positive integers, with $k \geq 2$, and \mathcal{G} an attributed graph. A collection M of sets of vertices is a CoHoP in a graph \mathcal{G} if and only if it satisfies the three following constraints:*

- $C_\alpha^{\text{hom}}(M) \equiv |\text{CATB}(M)| \geq \alpha$ (the vertices in M are homogeneous);
- $C_{\gamma,k}^{\text{kpc}}(M) \equiv |\mathcal{C}_{\text{kpc}}(\mathcal{G}[\cup_{C \in M} C])| \geq \gamma$, i.e., M contains at least γ k-PCs;
- $C^{\text{max}}(M) \equiv M = \mathcal{C}_{\text{kpc}}(\mathcal{G}[\text{CATB}(M)])$, i.e., M contains all k-PCs sharing the attributes in $\text{CATB}(M)$ and only these k-PCs.

Before comparing the constraints used to define MHCS and CoHoP patterns, let us give an example of CoHoP pattern.

Example 23 *In the attributed graph presented Figure 20, consider the CoHoP patterns satisfying constraints C_2^{hom} (i.e., all vertices share at least two attributes), $C_{2,3}^{\text{kpc}}$ (i.e., at least two 3-PCs) and C^{max} . The collection of set of vertices $\{ACD, DEFH, IJK\}$ satisfies these constraints. Indeed, all vertices share attributes a_4 and a_5 . Moreover, the collection is formed by three 3-PCs so $C_{2,3}^{\text{kpc}}$ is satisfied. Finally, there is no other 3-PC in the attributed graph induced by the attributes $\{a_4, a_5\}$, thus C^{max} is also satisfied.*

5.2 FINDING ALL COHOP PATTERNS

We first present a naive algorithm enumerating all subgraphs possibly containing a pattern. Then we show how we can safely reduce the subgraphs enumeration, and we describe the corresponding algorithm. Implementation techniques are discussed in Section 5.2.4.

5.2.1 A naive algorithm

While Definition 19 is very declarative, we establish a more constructive definition of the CoHoP patterns as follows.

Lemma 7 *Let k , α , and γ be three strictly positive integers, with $k \geq 2$, and \mathcal{G} be an attributed graph with \mathcal{A} the set of Boolean attributes in \mathcal{G} . A collection M of sets of vertices is a CoHoP if and only if there exists $X \subseteq \mathcal{A}$ such that $M = \mathcal{C}_{\text{kpc}}(\mathcal{G}[X])$, $|X| \geq \alpha$, and $|M| \geq \gamma$.*

Proof 8 *First, consider a CoHoP M . By direct application of Definition 19, there exists $X = \text{CATB}(M) \subseteq \mathcal{A}$ such that $M = \mathcal{C}_{\text{kpc}}(\mathcal{G}[X])$, $|X| \geq \alpha$, and $|M| \geq \gamma$. Now we prove the converse. Consider X a set of attributes satisfying $|X| \geq \alpha$, and M a collection of sets of vertices such that $M = \mathcal{C}_{\text{kpc}}(\mathcal{G}[X])$ and $|M| \geq \gamma$. Since $M = \mathcal{C}_{\text{kpc}}(\mathcal{G}[X])$, then $X \subseteq \text{CATB}(M)$, and each k-PC of $\mathcal{G}[\text{CATB}(M)]$ is included in or equal to a k-PC of $\mathcal{G}[X]$. Since all vertices in M are also in $\mathcal{G}[\text{CATB}(M)]$, then each k-PC member of the collection $M = \mathcal{C}_{\text{kpc}}(\mathcal{G}[X])$ is included in or equal to a k-PC of $\mathcal{G}[\text{CATB}(M)]$. Thus $M = \mathcal{C}_{\text{kpc}}(\mathcal{G}[X]) = \mathcal{C}_{\text{kpc}}(\mathcal{G}[\text{CATB}(M)])$ and M is a CoHoP.*

To compute all patterns, a naive algorithm can enumerate the subgraphs $\mathcal{G}_e = \mathcal{G}[\mathbf{X}]$ for all non empty sets of attributes X , and for each \mathcal{G}_e compute all k-PCs in \mathcal{G}_e . Then, if $|X| \geq \alpha$ and if there is at least γ k-PCs in \mathcal{G}_e , this collection of k-PCs is a CoHoP. From Lemma 7, this algorithm is correct. However, with this enumeration technique, $2^{|\mathcal{A}|} - 1$ subgraphs will have to be enumerated (there are $2^{|\mathcal{A}|} - 1$ non empty subsets of \mathcal{A}). The following pruning techniques are used to avoid the enumeration of some subgraphs.

5.2.2 Enumeration tree pruning techniques

In order to avoid the enumeration of subgraphs that do not contain CoHoP patterns, we propose four safe pruning techniques. For each of them, we show its safety and give an example.

PRUNING 1 This pruning techniques allows to reduce the set of vertices under consideration. Indeed, only vertices in a k-maximal-clique can form a pattern, so the other vertices can be discarded.

Lemma 8 *Let \mathcal{G} be an attributed graph. Only vertices in a k-maximal-clique of \mathcal{G} can form a CoHoP in \mathcal{G} or in any subgraph of \mathcal{G} .*

Proof 9 *Direct, since a vertex which is not in a k-maximal-clique cannot be in any k-PC.*

Example 24 *When extracting CoHoPs with $k = 3$, consider the subgraph $\mathcal{G}[\{\alpha_1, \alpha_2\}]$ in the attributed graph presented Figure 20. It is formed by the set of vertices $\{A, B, C, F, G\}$. In this subgraph, F and G are not in a 3-maximal clique and consequently cannot be in a 3-PC.*

PRUNING 2 From the following property, this pruning allows to stop the enumeration once a subgraph containing less than γ k-maximal-cliques is found.

Lemma 9 *Let \mathcal{G} be an attributed graph. If \mathcal{G} does not contains at least γ k-maximal-cliques, then neither \mathcal{G} nor any subgraph of \mathcal{G} can contain a CoHoP.*

Proof 10 *Let \mathcal{G} be an attributed graph having less than γ k-maximal-cliques. Since all k-cliques in a k-maximal-clique are in the same k-PC, then the number of k-maximal-cliques cannot be greater than the number of k-PCs. So, \mathcal{G} cannot contain γ k-PCs and thus cannot contain a CoHoP. The same holds for any subgraph of \mathcal{G} , since a subgraph of \mathcal{G} cannot contain more k-maximal-clique than \mathcal{G} .*

Example 25 *Consider the subgraph $\mathcal{G}[\{\alpha_1, \alpha_3\}]$ formed by the vertices F, G, and H in the attributed graph presented Figure 20. As this subgraph contains only one k-maximal-clique, if $\gamma \geq 2$ the enumeration of the subgraphs of $\mathcal{G}[\{\alpha_1, \alpha_3\}]$ can stop since the \mathcal{C}_{kpc} constraint would not be satisfied.*

PRUNING 3 This pruning is used to avoid the enumeration of an attribute shared by all the vertices in the graph under consideration.

Lemma 10 *Let \mathcal{G} be an attributed graph, X the set of attributes shared by all vertices in \mathcal{G} and $x \notin X$ an attribute having value True for all vertices in \mathcal{G} , then we have $\mathcal{G}[\mathbf{X} \cup \{x\}] = \mathcal{G}$.*

Proof 11 *Straightforward, since by definition all vertices share the attributes in $X \cup \{x\}$.*

Example 26 *Consider the subgraph $\mathcal{G}[\{a_1, a_4\}]$ in the attributed graph presented Figure 20. It is formed by the set of vertices $\{A, C, D, F, H\}$. As attribute a_5 has also value True for all these vertices, we have $\mathcal{G}[\{a_1, a_4, a_5\}] = \mathcal{G}[\{a_1, a_4\}]$.*

PRUNING 4 According to the following lemma, we can avoid the enumeration of graphs (and their subgraphs) if they are induced by sets of attributes shared by an insufficient number of vertices to contain a CoHoP.

Lemma 11 *Let \mathcal{G} be an attributed graph and x an attribute shared by less than k vertices in \mathcal{G} . Then, the graph $\mathcal{G}[\{x\}]$ and all its subgraphs cannot contain a CoHoP.*

Proof 12 *Since $|\text{VERT}(x)| < k$, $\mathcal{G}[\{x\}]$ and its subgraphs contains less than k vertices thus they cannot contain a k -clique and by extension a CoHoP.*

Example 27 *In the attributed graph presented Figure 20, consider the subgraph $\mathcal{G}[\{a_5\}]$. Among the vertices forming this subgraph, the attribute a_3 has value True only for two vertices, F and H. Consequently, for a value of $k \geq 3$, no subgraph of $\mathcal{G}[\{a_5\}]$ induced by the attribute a_3 can form a CoHoP pattern.*

5.2.3 Algorithm description

A recursive function FindCoHoP, that takes advantage of Pruning 1 to 4 is presented as Algorithm 11. The input of the algorithm for the first call is the whole attributed graph, *i.e.*, $\mathcal{G}_e = \mathcal{G}$, and \mathcal{A}_c , the set of candidate attributes that remain under consideration to find attributes shared by subgraph, is set to \mathcal{A} .

Line 1 checks that there is at least γ k -maximal-cliques in \mathcal{G}_e . If it is not the case, from Lemma 9 no subgraph of \mathcal{G}_e including \mathcal{G}_e itself can contain a k -PC. **Line 2** computes the set \mathcal{V}_r of vertices possibly containing a k -PC as the union of all k -maximal-cliques in \mathcal{G}_e according to Lemma 8. **Line 3** checks (1) if there is at least α attributes shared by all vertices in \mathcal{V}_r ($|\bigcap_{v \in \mathcal{V}_r} \text{ATB}(v)| \geq \alpha$) and (2) if there is at least γ k -PCs ($|\mathcal{C}_{k\text{PC}}(\mathcal{G}_e[\mathcal{V}_r])| \geq \gamma$). If so, the collection of k -PCs is a CoHoP, and is output on **line 4**. On **line 5**, attributes from \mathcal{A}_c shared by all vertices in \mathcal{V}_r are removed from \mathcal{A}_c . Removing these attributes does not change the collection of enumerated subgraphs, since if we pick such an attribute x we have $\mathcal{G}_e[\mathcal{V}_r \cap \text{VERT}(x)]$ that is equal to $\mathcal{G}_e[\mathcal{V}_r]$ itself in the recursive call to FindCoHoP (**line 9**). On **line 6**, attributes shared by less than k vertices in \mathcal{V}_r are removed from \mathcal{A}_c , according to Lemma 11. This avoids unnecessary calls to FindCoHoP with subgraphs having not enough vertices. **Lines 7 to 9** perform a standard recursive enumeration scheme to produce in a depth-first way, and element by element (the x that is picked), all subsets of \mathcal{A}_c . While \mathcal{A}_c is not empty, an attribute x is picked (**line 8**) and function FindCoHoP is called with the subgraph of \mathcal{G}_e induced by the set of vertices in \mathcal{V}_r sharing attribute x , *i.e.*, $\mathcal{G}_e[\mathcal{V}_r \cap \text{VERT}(x)]$.

Theorem 4 *Algorithm 11 returns all CoHoP patterns and only CoHoP patterns.*

Algorithm 11: FindCoHoP

Input: $\mathcal{G}_e, \mathcal{A}_c$

```

1 if  $|\mathcal{C}_{kmax}(\mathcal{G}_e)| \geq \gamma$  then // Pruning 2
2    $\mathcal{V}_r = \cup_{C \in \mathcal{C}_{kmax}(\mathcal{G}_e)} C$  // Pruning 1
3   if  $|\cap_{v \in \mathcal{V}_r} \mathbf{ATB}(v)| \geq \alpha$  and  $|\mathcal{C}_{kpc}(\mathcal{G}_e[\mathcal{V}_r])| \geq \gamma$  then
4     output  $\mathcal{C}_{kpc}(\mathcal{G}_e[\mathcal{V}_r])$ 
5      $\mathcal{A}_c \leftarrow \{x \in \mathcal{A}_c \mid \mathcal{V}_r \not\subseteq \mathbf{VERT}(x)\}$  // Pruning 3
6      $\mathcal{A}_c \leftarrow \{x \in \mathcal{A}_c \mid |\mathbf{VERT}(x) \cap \mathcal{V}_r| \geq k\}$  // Pruning 4
7     while  $\mathcal{A}_c \neq \emptyset$  do
8       Pick and remove an attribute  $x$  from  $\mathcal{A}_c$ 
9       FindCoHoP( $\mathcal{G}_e[\mathcal{V}_r \cap \mathbf{VERT}(x)]$ ,  $\mathcal{A}_c$ )

```

Proof 13 Lemma 7 and Lemmas 8 to 11 (safety of the pruning) ensure the completeness of Algorithm 11. Line 3 ensures its soundness.

Note that a given CoHoP might be output several times by Algorithm 11. Such duplicates are removed in a simple post-processing step.

5.2.4 Implementation

Since vertices in a pattern must share at least one attribute ($\alpha \geq 1$), usually it is not necessary to compute the k -maximal-cliques on the whole attributed graph. So, the first level of the enumeration is computed using only lines 6 to 9 of Algorithm 11, with \mathcal{V}_r the set of all vertices of the input attributed graph.

The algorithm used to compute the collection of k -PCs in a graph is described as Algorithm 3 in Section 2.2.3.1 of the state of art [29, 67]. It first builds a matrix representing the adjacency relation between the k -cliques, then computes the connected components from this matrix, which are the k -PCs. The algorithm used to compute the k -maximal-cliques is CLIQUES [84] described as Algorithm 2 in Section 2.2.1.1 of the state of the art. Both the collection of k -maximal-cliques (i.e., $\mathcal{C}_{kmax}(\mathcal{G}_e)$) and the collection of k -PCs (i.e., $\mathcal{C}_{kpc}(\mathcal{G}_e[\mathcal{V}_r])$) are computed only once for a given attributed graph \mathcal{G}_e on respectively lines 1 and 3. Further use of $\mathcal{C}_{kmax}(\mathcal{G}_e)$ and $\mathcal{C}_{kpc}(\mathcal{G}_e[\mathcal{V}_r])$ on lines 2 and 4 use the collections previously computed. Moreover, the computation of the k -PCs is done on line 3 only if the vertices in \mathcal{V}_r share at least α attributes (i.e., $|\cap_{v \in \mathcal{V}_r} \mathbf{ATB}(v)| \geq \alpha$).

5.3 EXPERIMENTS

In this section we report experiments on the bibliographic datasets already presented in the Section 4, DBLP₁, DBLP₂, and DBLP₃.

First, we present two examples of CoHoP patterns found in DBLP₃. Next, we present and discuss the performances of the algorithm. The impact of several graph characteristics over runtime is studied using a synthetic dataset. Finally we present the runtime improvement for each pruning technique with respect to a baseline algorithm.

All experiments were performed on a PC running GNU/Linux with a 3 GHz Core 2 Duo CPU and 8 GB of main memory installed. The

α	γ	k	# CoHoP
5	7	3	51
3	7	4	57
4	9	3	79
3	6	4	84
5	6	3	85

Table 4.: The five set of parameters with α between 3 and 9, γ between 5 and 9, and k between 3 and 6 that lead to a number of CoHoPs between 50 and 100 in DBLP₃.

algorithm has been implemented using Scala 2.9. This algorithm has also been embedded in the software presented in Appendix A.

5.3.1 Illustration of the patterns interest

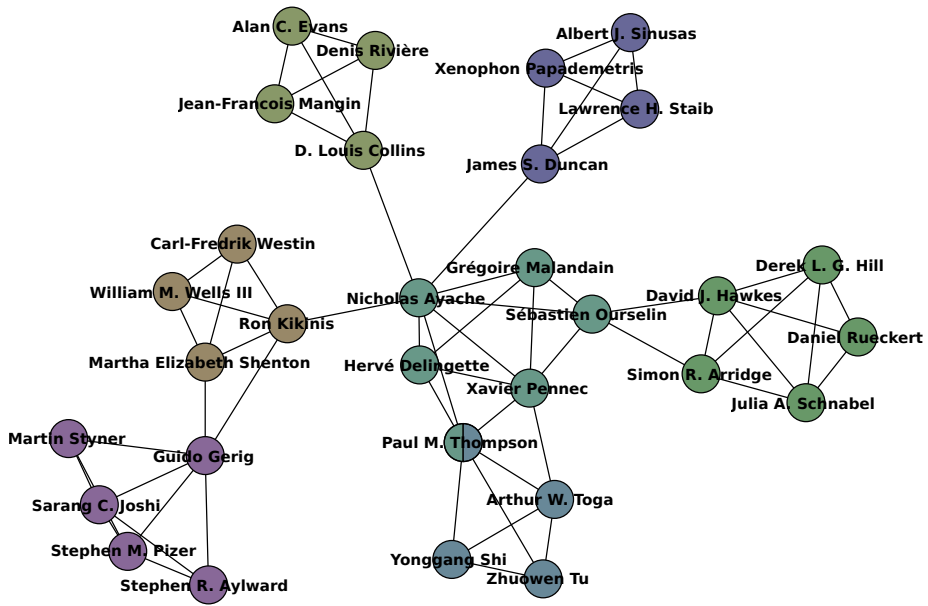
Let us first define some vocabulary in the context of a network of researchers. In [67] the authors consider that a k -PC is a *community* in the sense that “it consists of several complete subgraphs that tend to share many of their nodes”. Consequently, we will use the term community for a k -PC. We will also say that two communities are connected if there is an edge between both communities.

To set the parameters α , γ , and k we wanted between 3 and 9 shared attributes and a value of γ between 5 and 9. Concerning parameter k , in [29, 67, 68] the authors advice to use a value between 3 and 6. Moreover, we wanted a relatively small collection of CoHoP patterns but still containing different structures, so we required a collection containing 50 to 100 CoHoPs. Table 4 presents the five parameters settings, within the parameters ranges given above, that satisfy these constraints in DBLP₃. Among these parameters, we selected $k = 4$, $\gamma = 7$, and $\alpha = 3$, *i.e.*, CoHoPs with at least seven 4-PCs where all authors have published in the same three conferences or journals.

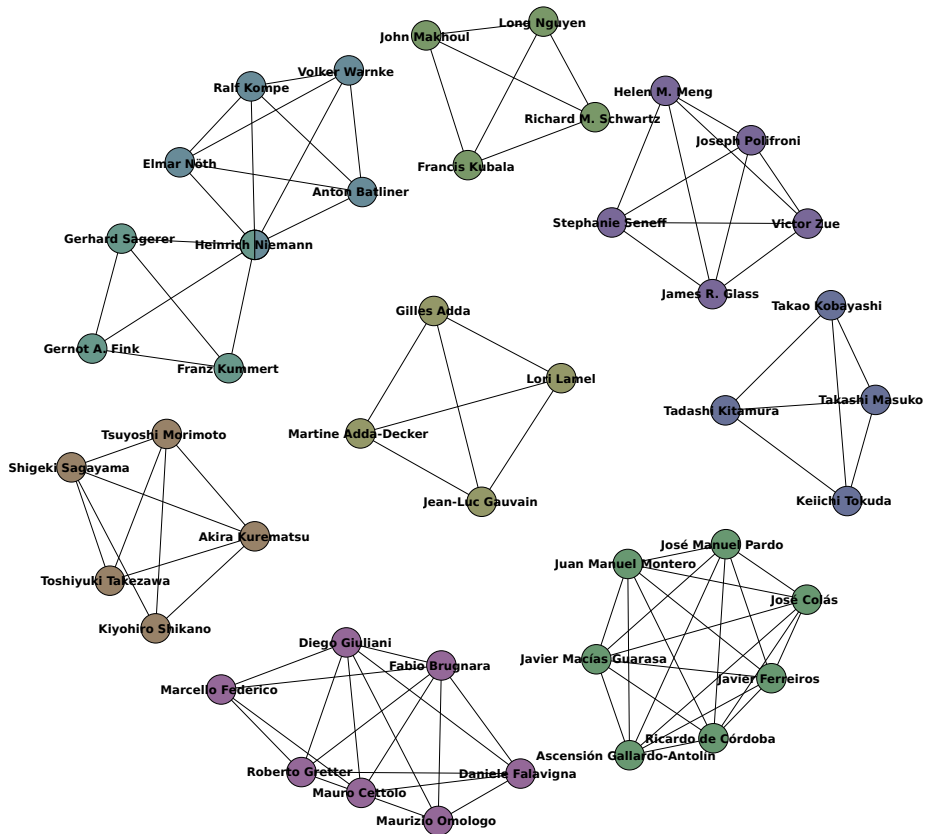
With this parameters setting, 57 CoHoPs were extracted. To illustrate the kind of patterns that were retrieved, we focus on two patterns presented in Figures 21a and 21b.

The pattern in Figure 21a contains seven 4-PCs, all authors having published in conferences or journals related to medical imaging. The authors N. Ayache, H. Delingette, G. Malandain, S. Ourselin, X. Pennec, and P. M. Thompson form a community connected to all other communities except one and is the core of a star-based topology. Knowing such a structure is useful to make some decisions. For instance having researchers of the core community as partners in a project, or choosing this community as a destination for a post-doc position could be a great opportunity to benefit from contacts with all the other groups. We also investigated the role of the authors connecting two communities (*i.e.*, the endpoints of edges connecting two communities) in this pattern using ArnetMiner. We found that four of these *bridging nodes* [64] were advisers of at least half of the authors of their respective communities. So they are likely to be senior researchers and this is coherent with the fact that they appear as bridges between communities.

ArnetMiner
(<http://arnetminer.org/>)
is an application
providing the
relationship (e.g.,
coauthor, adviser,
advisee) between
researchers.



(a) Seven 4-PCs concerning conferences IPMI, ISBI, MICCAI and journal IEEE Trans. Med. Imaging.



(b) Nine 4-PCs concerning conferences INTERSPEECH, ICSLP, and EUROSPEECH.

Figure 21.: Two patterns extracted from DBLP₃ with $k = 4$, $\gamma = 7$, and $\alpha = 3$. Each colour corresponds to a k-PC. A vertex in several colours is contained in multiple k-PCs.

	DBLP ₁	DBLP ₂	DBLP ₃
Mean	1,298 Mb	507 Mb	243 Mb
Max	2,264 Mb	884 Mb	489 Mb
Standard deviation	235 Mb	140 Mb	61 Mb

Table 5.: Memory consumption over all experiments reported in Figure 22.

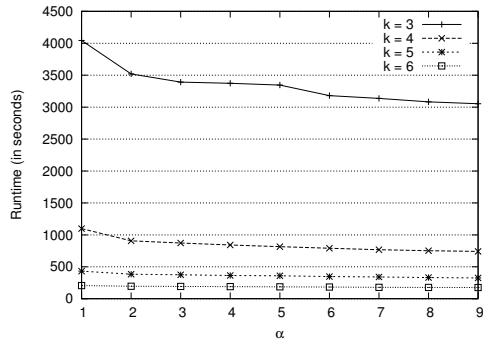
In the second CoHoP, presented Figure 21b, all authors have published at least three times in three conferences related to speech communication / spoken language (EUROSPEECH, INTERSPEECH, ICSLP). It contains nine communities, seven of them not being connected to any other. It is interesting to notice that EUROSPEECH and ICSLP were two conferences organized over a decade from 1990 to 1999 and merged in 2000 giving rise to the INTERSPEECH conference (still active in 2011). This indicates that the part of the research activity denoted by the pattern is very homogeneous in terms of research domain. In addition, since all researchers of the pattern have published at least three times in EUROSPEECH, ICSLP and INTERSPEECH, this means that the pattern depicts an activity over two decades, and that these researchers are likely to have been active in the domain over a rather long period. Moreover, from the personal pages of the authors, we found out that in most cases a community is formed by people working in the same research institute. So, here most communities are formed by researchers working in the field of speech processing and not strongly publishing with researchers from other institutes. Such structure with disconnected groups of people sharing similar interest might be interesting for several tasks. For instance, it can give hints to funding agencies to set up long term development strategies of collaboration networks. It can also be helpful, in a normal day-to-day activity, like finding reviewers for a article, by suggesting experts in the same domain as the authors, but having no closed collaborations (no strong coauthor relationship) with these authors, and also eventually having no closed collaborations with the other experts (picking them in other disconnected groups).

5.3.2 Performance study

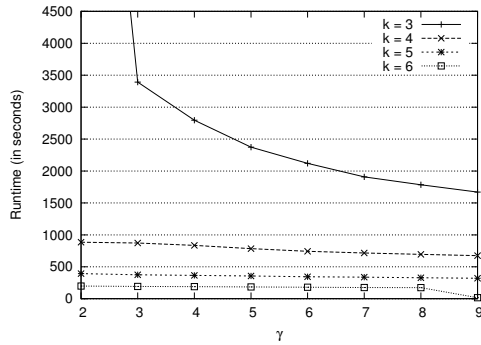
Concerning time performances, Figure 22 show that the extraction can be made in less than 20 minutes when $k \geq 4$ on the three datasets. Indeed, the extraction requires less than a few tens of seconds on DBLP₂ and DBLP₃ for all recommended k values (between 3 and 6). On the DBLP₁ dataset (using the complete DBLP database) the extractions can take several thousands of seconds, but remain feasible. We can also notice that, as expected, for weaker constraints (lower values of α , γ , and k) the runtime increases.

Regarding the number of output patterns, Figure 23 shows that it shrinks fast when parameter values increase (*i.e.*, stronger constraints). In particular, when k increases by two, the size of the collection of patterns decreases by more than one order of magnitude in all settings.

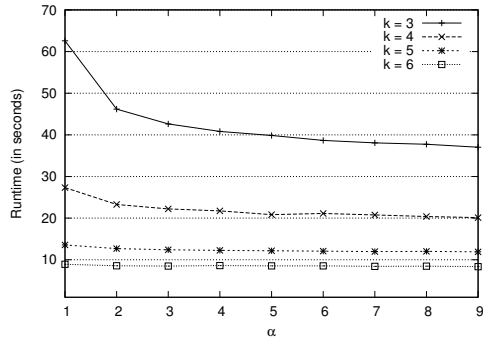
For the main memory usage, Table 5 reports the maximal memory consumption during each extraction reported in this section.



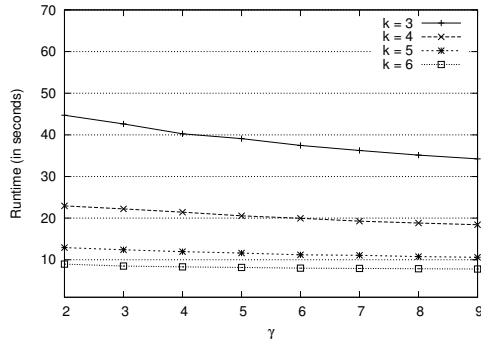
(a) Runtimes on DBLP₁ with $\gamma = 3$



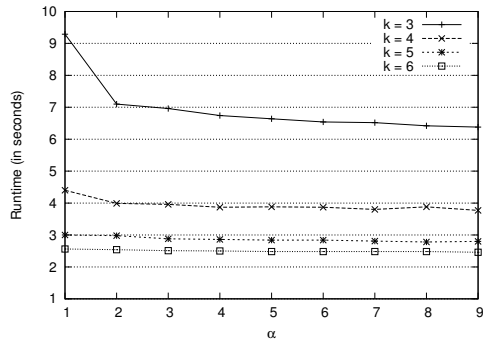
(b) Runtimes on DBLP₁ with $\alpha = 3$



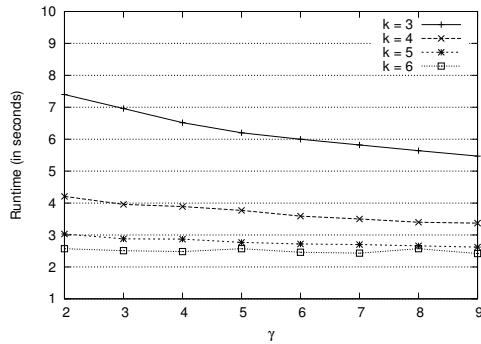
(c) Runtimes on DBLP₂ with $\gamma = 3$



(d) Runtimes on DBLP₂ with $\alpha = 3$

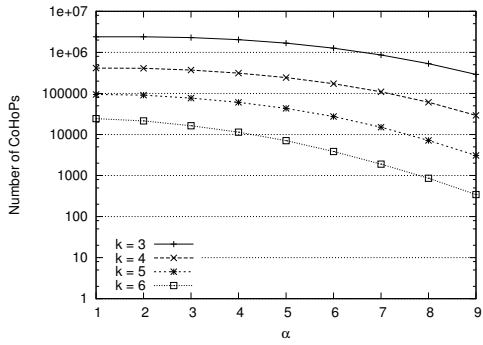


(e) Runtimes on DBLP₃ with $\gamma = 3$

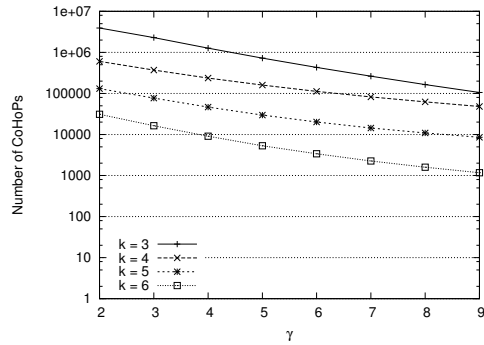


(f) Runtimes on DBLP₃ with $\alpha = 3$

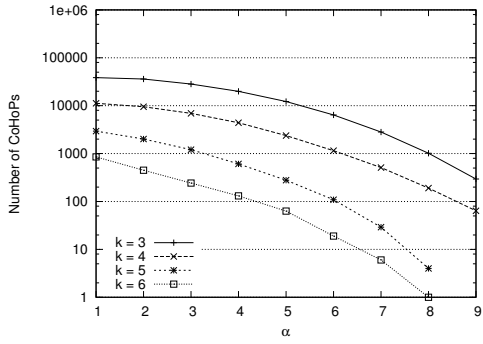
Figure 22.: Runtime for different sets of parameters on DBLP₁, DBLP₂, and DBLP₃.



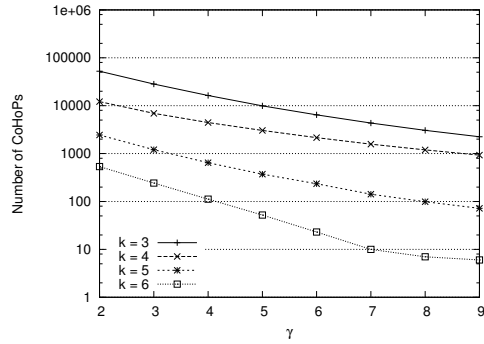
(a) # CoHoP on DBLP₁ with $\gamma = 3$



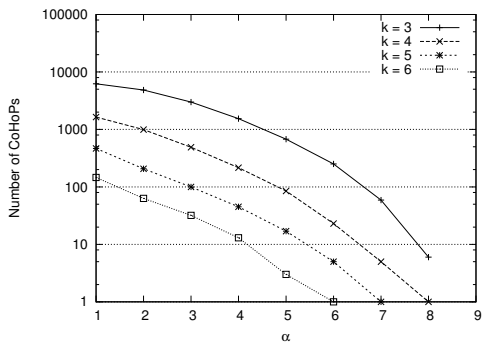
(b) # CoHoP on DBLP₁ with $\alpha = 3$



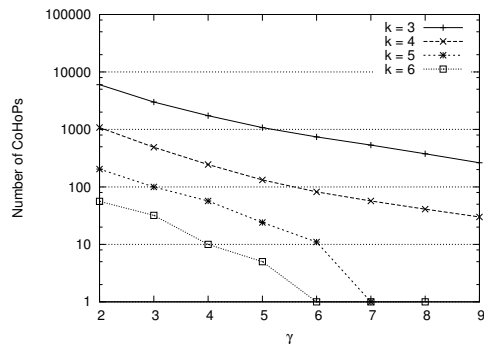
(c) # CoHoP on DBLP₂ with $\gamma = 3$



(d) # CoHoP on DBLP₂ with $\alpha = 3$



(e) # CoHoP on DBLP₃ with $\gamma = 3$



(f) # CoHoP on DBLP₃ with $\alpha = 3$

Figure 23.: Number of CoHoP for different sets of parameters on DBLP₁, DBLP₂, and DBLP₃.

	$S_{\#vert}$	$S_{\#attr}$	S_{avgDeg}	S_{avgAtt}	$S_{\#CoHoP}$
#vert	5,000-25,000	15,000	15,000	15,000	15,000
#attr	500	200-800	500	500	500
avgDeg	20	20	10-30	20	20
avgAtt	10	10	10	5-25	10
#CoHoP	300	300	300	300	100-500

Table 6.: Parameters used to generate the synthetic datasets.

5.3.3 Evaluation on synthetic datasets

In this section we describe an experimental evaluation of the algorithm using synthetic datasets as made for the MHCSs in Section 4.3.4.

The synthetic datasets generator and the generation parameters are the same as the ones used in the experiments for MHCS presented in Section 4.3.4 except for the injected patterns. Here the number of patterns is #CoHoP and three parameters control their structure: s , l , and g .

All the injected CoHoP are structured such that they are formed by g l -PCs. A l -PC is built by inserting g overlapping l -cliques in the graph. Each l -clique share at most $l - 1$ vertices with the other l -cliques. More precisely, the l -cliques are built from a previous l -clique (except the first one which is picked at random) by removing a vertex and adding a new vertex not present in any other l -clique forming the injected CoHoP. The vertices forming the CoHoPs were then randomly associated to s attributes. Table 6 summarizes the parameters settings used for all synthetic datasets. These settings are similar to the ones used for the MHCSs. From each random dataset, we obtained four derived datasets by injecting four sets of random CoHoPs obtained with the four following settings: (1) $s = 2$, $l = 6$, and $g = 2$, (2) $s = 2$, $l = 6$, and $g = 4$, (3) $s = 4$, $l = 6$, and $g = 2$, and (4) $s = 4$, $l = 6$, and $g = 4$. As for the experiments on the MHCSs, we used s , l , and g as values for the extraction parameters and set $\alpha = s$, $k = l$, and $\gamma = g$. The corresponding runtimes are given Figure 24 where each point is the average runtime of extractions over ten different random datasets.

These results show that the runtime scales well with respect to the parameters of the attributed graph generation: number of vertices, attributes and CoHoPs, average vertex degree and average number of attributes with value True per vertex.

5.3.4 Comparison with baseline algorithm

Here we study the gain of each pruning techniques regarding the runtime. We proposed four versions of the algorithm based on a baseline algorithm corresponding to the naive version proposed in Section 5.2.1. As for the MHCSs, we incrementally added the pruning techniques to the baseline algorithm, starting from Pruning 1 to Pruning 4. The algorithm version using prunings 1+2+3+4 is then the same as the one presented as Algorithm 11.

The experiments were only run over DBLP₂ and DBLP₃ since extraction runtimes using only Pruning 1 were prohibitive on DBLP₁. For the

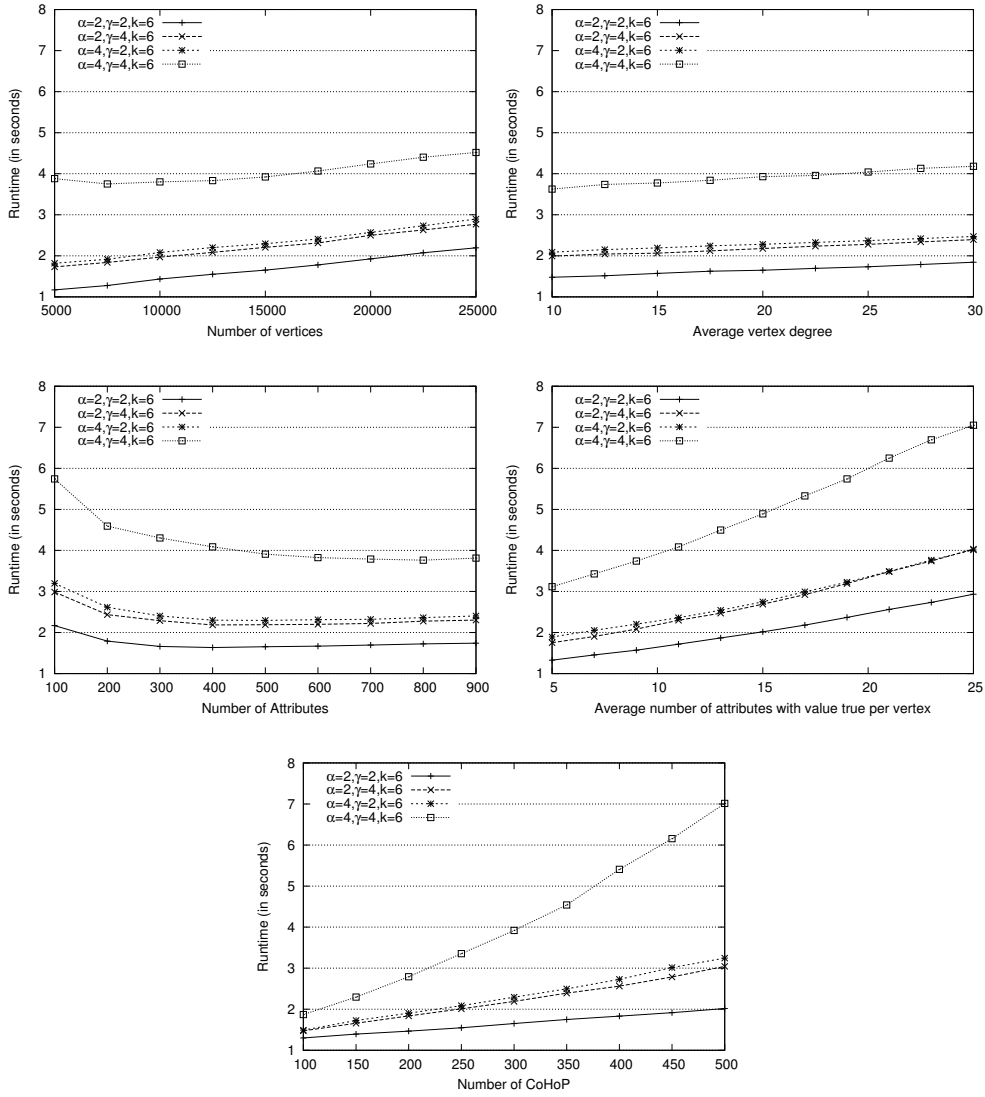
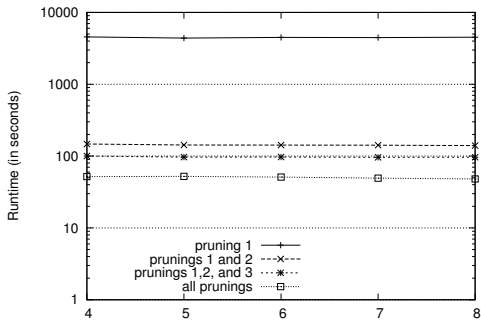
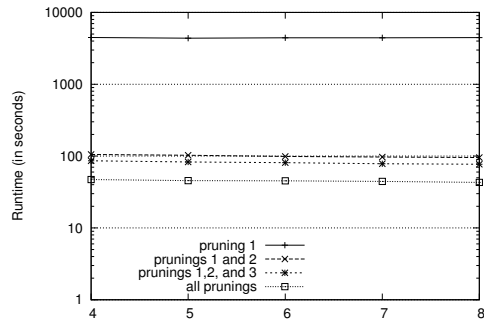


Figure 24.: Runtimes for the extraction of CoHoP patterns in the collections of datasets $S_{\#vert}$, S_{avgDeg} , $S_{\#attr}$, S_{avgAtt} and $S_{\#CoHoP}$.

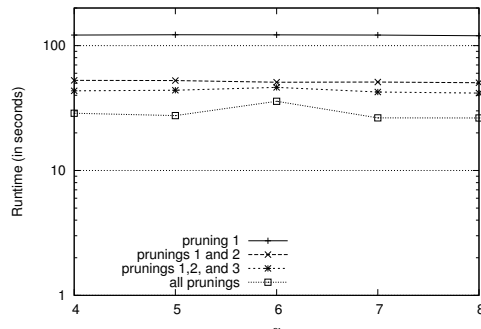
same reason, the runtimes without any pruning were not collected. The results are presented in Figure 25. We observe that using incrementally the Pruning 1 to 4 improves significantly the runtime in most cases.



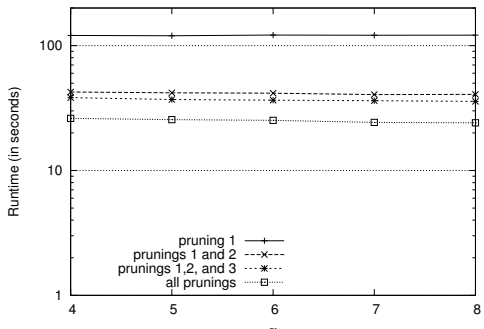
(a) DBLP₂ with $\gamma = 2$ and $k = 3$.



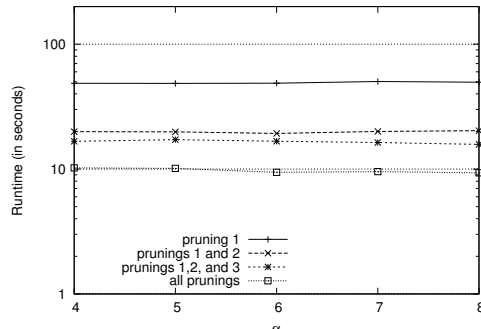
(b) DBLP₂ with $\gamma = 3$ and $k = 3$.



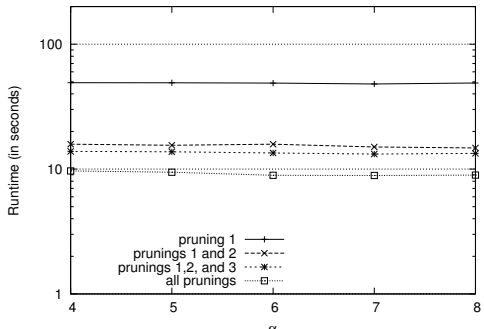
(c) DBLP₂ with $\gamma = 2$ and $k = 4$.



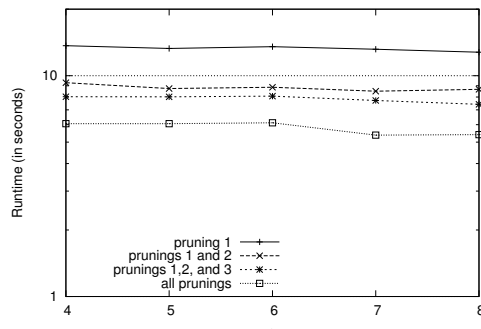
(d) DBLP₂ with $\gamma = 3$ and $k = 4$.



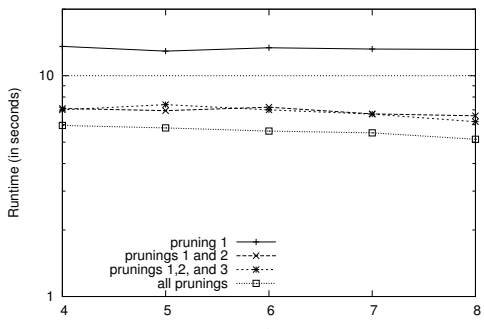
(e) DBLP₃ with $\gamma = 2$ and $k = 3$.



(f) DBLP₃ with $\gamma = 3$ and $k = 3$.



(g) DBLP₃ with $\gamma = 2$ and $k = 4$.



(h) DBLP₃ with $\gamma = 3$ and $k = 4$.

Figure 25.: Runtime for the extraction of CoHoP patterns using different pruning techniques on DBLP₂ and DBLP₃. The scale is logarithmic for the runtime.

CONCLUSION

In this section we presented the main contributions of this thesis. In the Boolean attributed graph setting, we defined two classes of patterns in order to find sets of homogeneous groups, namely, the Maximal Homogeneous Cliques Sets (MHCS) and the Collections of Homogeneous k -clique Percolated components (CoHoP).

Both the MHCSs and the CoHoPs are based on constraints specifying a minimum number of shared attributes (the homogeneity) and a minimum number of large groups forming a pattern. However, their definitions differ in several points. These classes of patterns require a different topology for the groups, a MHCS is formed by complete subgraphs while a CoHoP is made of k -clique percolated components. Moreover, a CoHoP is formed only by groups containing at least k vertices, whereas a MHCS can collect small additional contextual groups (*i.e.*, groups containing less than k vertices). The definition of maximality used for the MHCSs also requires that a MHCS cannot be included in another MHCS.

Notice that the cliques of a MHCS M obtained with parameters α , γ and κ cannot always be used to build a CoHoP satisfying parameters α , γ and $k = \kappa$. This is because two cliques of M containing more than κ vertices can be merged (if they shared at least k vertices) in a single k -PC. So, it is possible that the subgraph induced by the vertices in M contains less than γ k -PCs, and cannot form a CoHoP. Thus, there is no direct relationship between the number of MHCSs and the number of CoHoPs. Depending on the data and for a parameter setting α , γ and $k = \kappa$, we can obtain more MHCSs or more CoHoPs.

In order to compute the collections of patterns, we proposed two sound and complete algorithms. These algorithms reduce the search space by taking advantage of several pruning techniques, for which we prove the safety.

The performance of these algorithms is studied using large real and synthetic datasets. The results show that both approaches scale well with respect to different attributed graph characteristics. We also presented several examples of usage for both classes of patterns for the analysis of a network of scientific collaborations.

Beyond the design of the algorithms and their implementation as research prototypes, we also developed a fully fledged software tool in order to ease the visualisation and browsing of the MHCSs and CoHoPs (software presented in Appendix B).

Part IV

APPLICATION TO MOLECULAR BIOLOGY

INTRODUCTION

The accumulation of large protein-protein interaction (PPI) networks and the need for their analysis has led to the emergence of new research domains in systems biology. Recently, the simultaneous use of both PPI networks and gene related properties such as biological functions [69], species [49] or expression in biological situations [86] has provided biological contexts from which meaningful structures can be extracted. When using both PPI networks and the biological functions associated to genes, an important number of methods have been proposed to extract *functional modules* (a short overview is proposed in [86] and [94]). They are usually defined as a group of cellular component (*e.g.*, proteins) and their interactions that can be associated to a specific biological function [39].

Our application, made in collaboration with an expert in biology (Dr. Olivier Gandrillon¹), draws its motivation from the result in [89] stating that it is not possible to infer the PPI network using only gene co-expression data. Since genes co-expressed in the same set of biological situations do not necessarily produce proteins in interaction, one can deduce that there might be several groups of interacting proteins, or *protein modules* whose corresponding genes are overexpressed in the same biological situations. Such groups might have strong inner interaction, and few or no interactions with proteins in other protein modules. Then, when analysing a protein module, it is possible to get a larger picture by retrieving the other modules of proteins produced by genes involved in the same set of biological situations. Studying such structures might also be useful when studying biological questions regarding modular cell biology as proposed in [39]. Such questions might be, for instance: Why protein modules overexpressed in similar situations are disconnected? How do connections between modules change during evolution?

Given this context, the MHCS and CoHoP patterns fit well for the analysis of such data. The proteins are encoded as vertices, and their interactions as edges. Boolean attributes, representing the biological situations, are associated to each protein. These attributes encode the fact that the protein is considered to be overexpressed or not in a given biological situation. In such setting, a MHCS or a CoHoP is a collection of protein modules where all proteins correspond to genes that are overexpressed in a same set of biological situations.

In the following sections, we report experimental results in two biological datasets using both MHCS and CoHoP patterns. We will use the term pattern to refer to both MHCSs and CoHoPs. After a description of the datasets we propose a measure based on a p-value to assess the biological interestingness of the extracted patterns. Then, we focus on four patterns, two MHCSs and two CoHoPs, and we provide a biological interpretation of these patterns. Finally, we present quantitative results regarding the runtime and the number of output patterns.

1. Centre de Génétique et de Physiologie Moléculaire et Cellulaire, CNRS UMR 5534, Villeurbanne.

6.1 THE DATASET

We built two datasets, BioData₁₅₀ and BioData₄₀₀, by processing and merging information from two databases: STRING¹ [42] and SQUAT² [48]. STRING integrates data on protein-protein interactions from different sources (*e.g.*, genomic data, co-expressions, other wet experiments, literature). Evidences of interaction are weighted depending on the source, and a confidence score between 150 and 1000 is given. The dataset BioData₁₅₀ have been built using all reported interactions (*i.e.*, with a confidence threshold of 150). The second dataset, BioData₄₀₀, uses all interaction with a confidence higher or equal to 400, that is the default STRING selection threshold. For the sake of model simplicity, we kept only proteins for which correspond a unique gene encoding the protein (this is the case for most of the proteins). So, the graph can also be interpreted as a gene-gene interaction graph, and in the following we will use both the terms genes and proteins to refer to the vertices.

SQUAT is a public database of Boolean gene expression data resulting from SAGE experiments [87]. A complete description of the discretization process is available in [6].

Both databases follow the HUGO³ nomenclature (see Figure 26) to encode gene names, so this nomenclature was used to link them. In our experiments, only Human species genes and their corresponding proteins were used. For these genes we have expression data in SQUAT for 486 biological situations. Several measures describing the datasets are presented in Table 7. A more complete description of the data is available in Appendix A.

The qualitative experiments were performed using the BioData₄₀₀ since we want to consider only the interactions between the proteins for which we have a relatively high confidence. The BioData₁₅₀ dataset is used for the quantitative experiments in order to compare the performances with BioData₄₀₀ which have a lower average degree.

6.2 EVALUATION OF THE PATTERN COLLECTION

6.2.1 *The L2L Measure*

An important difficulty of experiments on real datasets is the qualitative evaluation of the whole collection that is output, without having to ask to an expert to interpret/to assess the patterns one by one. Different statistical approaches exist to measure pattern significance, however in most cases they do not take in account domain information. In the case of the patterns extracted from the bibliographic dataset, we are not aware of typical tool to assess such collections, but fortunately, for

1. <http://string-db.org/>, snapshot of November 2009.

2. <http://bsmc.insa-lyon.fr/squat/>, snapshot of November 2009

3. <http://www.genenames.org/>

Query results

Gene name

Query summary :

Number of libraries found : 24 (4.93% of all libraries)

Query parameter : gene product **CRX** (Homo sapiens cone-rod homeobox , mRNA)

Tag : CAAGGTCTG   - Threshold : *unspecified*

All expression values are expressed in tags per million.

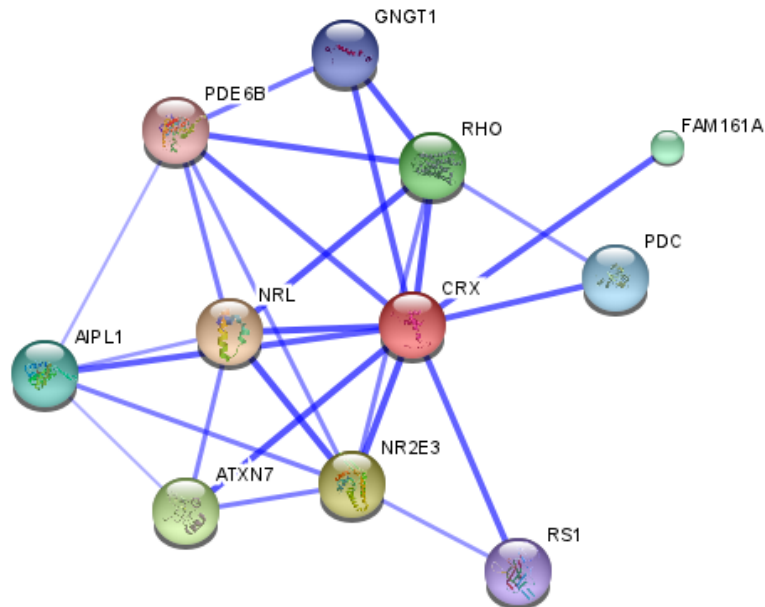
[See libraries where tag is not expressed](#)

Biological Situations

Libraries :

- **Library 1991** : SAGE_Retina_Peripheral_normal_B_4Peri
Keywords : retina bulk: normal peripheral retina retina, normal peripheral retina, norm
Tag CAAGGTCTG expression value : 505
- **Library 1992** : SAGE_Retinal_Pigment_Epithelium_normal_B_4PeriRPE
Keywords : retina bulk: retinal pigment epithelium (RPE) and choroid retina, retinal pig
Tag CAAGGTCTG expression value : 288
- **Library 1989** : SAGE_Retinal_Pigment_Epithelium_normal_B_4MacRPE
Keywords : retina bulk: macular retinal pigment epithelium (RPE) and choroid retina, r
sage
Tag CAAGGTCTG expression value : 259

(a) Screen copy of the SQUAT access web site. It represents three biological situations, termed *libraries*, where the gene CRX is overexpressed.



(b) A protein-protein interaction network containing gene CRX from the STRING access web site.

Figure 26.: Screen copy from SQUAT and STRING databases, showing information about gene CRX (HUGO nomenclature).

	BioData ₁₅₀	BioData ₄₀₀
# Vertices	15,571	15,571
# Attributes	486	486
# Edges	458,713	155,784
Avg. degree	58.92	20.01
Maximum degree	1500	496
Avg. attributes/vertex	11.46	11.46
Maximum attributes	92	92

Table 7.: Several characteristics of the datasets BioData₁₅₀ and BioData₄₀₀.

sets of genes, as the ones contained in the MHCs and CoHoPs found in the biological dataset, some methods have already been designed.

So, it is possible to performed such systematic evaluation on complete collections of patterns for various parameter settings. To this aim, we used the p-value measure as computed by the L2L method [65], a well known tool designed to facilitate the interpretation of microarray experiments results. The measure is built using predefined list of genes, where each list is a list of genes known to be involved in a similar biological process (*e.g.*, transcription, biological regulation, vision).

The L2L p-value for a set of genes G and an L2L list of genes L is the probability to obtain the overlap of size at least $|G \cap L|$ if the elements in L were chosen uniformly at random (the null hypothesis). It is computed with the following cumulative binomial distribution, as recommended in [65]:

$$\text{p-value} = 1 - \sum_{x=0}^{q-1} \binom{n}{x} p^x (1-p)^{n-x}$$

The number of trials, n , is the number of genes in L ; the number of success, q , is the number of genes of L contained in G ; and for one trial the probability of success, p , is the number of genes in G divided by the number of different genes in the union of the lists (13,746 genes⁴).

For a given pattern, we took as set G the set of all genes in this pattern. Then, among all L2L lists, we retained the list, noted L_G , that led to the lowest p-value with respect to G . So, L_G was the list such that the observed intersection $G \cap L_G$ was the less likely to occur by chance. The pattern was then simply associated to this L2L list L_G and to the corresponding p-value. We set a base significance level of 0.05. Multiple hypothesis testing (one test against each L2L list) increased the odds to have a low p-value by chance. So, we adjusted the significance level using the common Bonferroni correction (simply dividing the base significance level by the number of tests). Since we used the 2,075 lists of L2L being related to biological functions, this correction resulted in a significance threshold of $0.05/2075 \approx 2.4 \times 10^{-5}$. Then, for a pattern associated to a list L_G , if the corresponding p-value was lesser than the adjusted significance threshold, we considered that the pattern was significantly related to the biological function associated to L_G in L2L.

4. We used a snapshot of the L2L lists based on the biological process reported in Gene Ontology. It contains 2,075 lists corresponding to a total of 13,746 different genes.

6.2.2 Global evaluation of complete collections of patterns

The global evaluation of the collection of patterns has been performed using BioData₄₀₀ in order to consider only the protein interactions for which we have a high confidence. Figure 27 and 28 present cumulative distributions of the number of respectively MHCS and CoHoP patterns extracted for different values of α , γ , κ , and k , according to their associated p-value. For most parameter settings (16 out of 20), at least 70 % of the patterns correspond to a p-value lesser than the adjusted significance threshold. The four cases that do not lead to such promising collections of patterns are among the weakest constraint settings. Indeed, this is observed for the MHCSs (see Figure 27), when $\kappa = 2$ and for the CoHoPs (see Figure 28), when $k = 2$ or $\alpha = 2$ or $\gamma = 2$. We can notice that when the values of the parameters increase (*i.e.*, the selection constraints are stronger), this also increases most of the times the percentage of patterns with a p-value below the significance threshold (*i.e.*, patterns significantly related to a biological function by the L2L measure). This is an interesting evidence, advocating that these constraints are appropriate and meaningful in this application.

6.2.3 Interpretation

6.2.3.1 Analysis of two MHCSs

The previous experiments, run to assess whole sets of extracted patterns, helped us then to set the parameters to find a meaningful and easy to browse collection (*i.e.*, rather small and containing patterns having low p-values). As shown Figure 27, three collections were such that 90% of the MHCSs have a p-value lesser than or equal to 10^{-10} . So, we choose to focus on one of these collections, and selected the one obtained for the thresholds $\alpha = 3$, $\kappa = 4$ and $\gamma = 3$. It contained 10 MHCSs, and here we present two of them.

The first MHCS is depicted Figure 29. For the sake of readability, Figure 29a shows the core cliques only (cliques having at least κ vertices), and all interactions (all cliques excluding isolated vertex) are given Figure 29b. This MHCS contains 17 cliques of size at least 4, and all genes are overexpressed in 3 biological situations that correspond to normal activities of retinal cells. This MHCS has the lowest L2L p-value (1.3×10^{-25}) of the whole collection, and this p-value corresponds to the L2L list entitled *sensory perception of light stimulus*⁵. This advocate for the biological coherence of the pattern since the common biological situations are retinal cell activities. With respect to the structure of the interaction graph, the MHCS reveals in Figure 29a that the core cliques are connected together, in particular through genes CRX and RHO, as confirmed in the Table 29c where it can be verified that CRX and RHO belong to all core cliques, except one. More global information about the structure are given Figure 29b that contains all interactions in the pattern and shows that most of these interactions occur within the core cliques (63 edges out of 87).

Additionally, the Table 29c also gives a finer grain L2L scoring of the core part of the pattern. For each core cliques (column 3), its own L2L measure has been computed and the table reports the retained

5. Sensory perception of light stimulus is defined as the series of events required for an organism to receive a sensory light stimulus, convert it to a molecular signal, and recognize and characterize the signal.

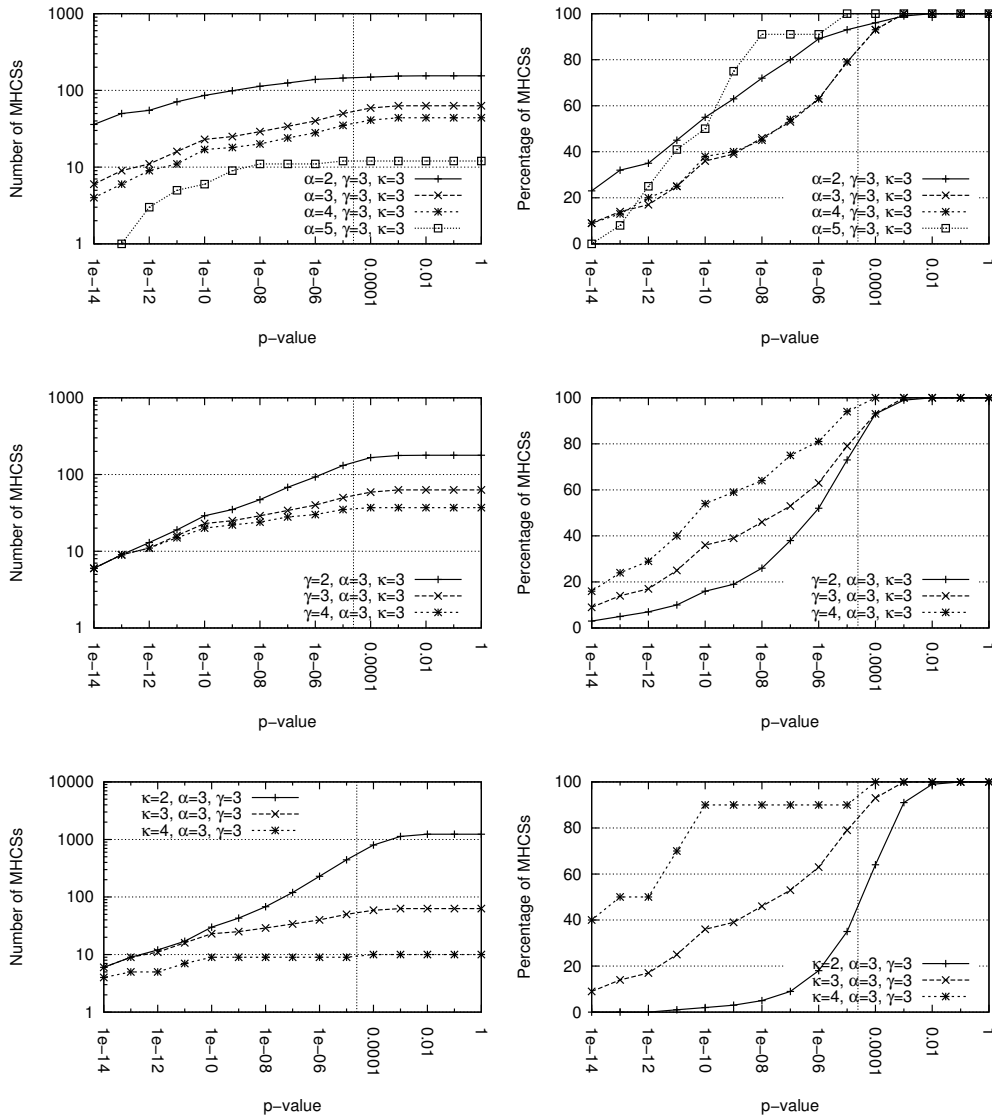


Figure 27.: Cumulative distributions of p-values (on dataset BioData₄₀₀ with α , γ and κ varying) in number of MHCs extracted (left column) and in percentage of the MHCs extracted (right column). P-value scale is logarithmic. The vertical dotted line corresponds to the significance level using Bonferroni correction (*i.e.*, $\approx 2.4 \times 10^{-5}$).

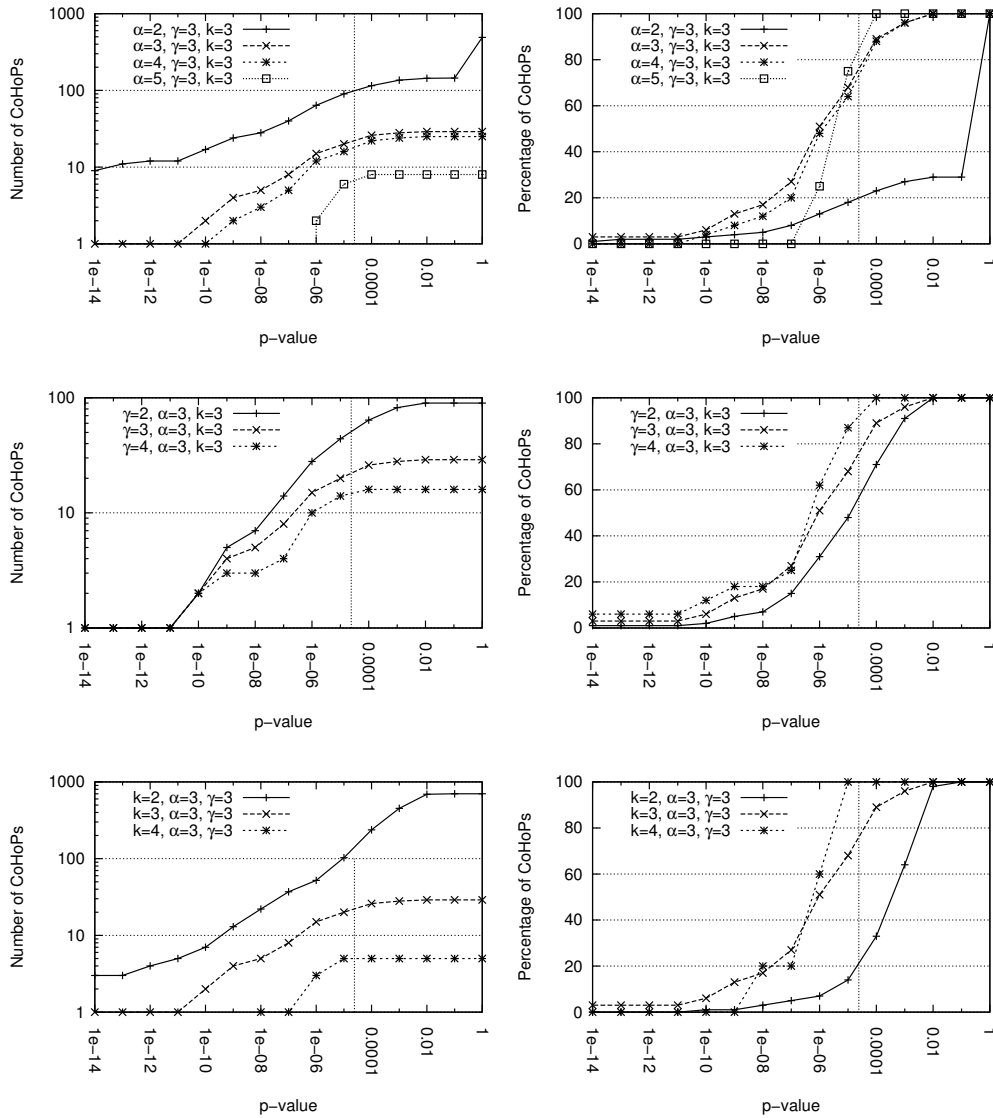


Figure 28.: Cumulative distributions of p-values (on dataset BioData₄₀₀ with α , γ and κ varying) , in number of CoHoPs extracted (left column) and in percentage of the CoHoPs extracted (right column). P-value scale is logarithmic. The vertical dotted line corresponds to the significance level using Bonferroni correction (*i.e.*, $\approx 2.4 \times 10^{-5}$).

p-value (column 1) and the name of the corresponding L2L list (column 2). In this table, the name *perception of light* is simply a short name used for *sensory perception of light stimulus*. As mentioned above, this biological process was associated to the whole pattern, but the table shows that it is also associated to most of the core cliques of the pattern in isolation. For the other core cliques, two are associated to the more specific process of *phototransduction* and the last one is associated to the more general process of *sensory perception*.

A different kind of structures is illustrated by a second MHCS presented Figure 30. Here, the core cliques of the patterns, depicted Figure 30a, show that the pattern is structured around two separated components (no direct interaction between the genes of these two groups), and the context, given Figure 30b, points out that most of the interactions are not in the core cliques (35 edges out of 98).

Additionally, from a structural point of view, Figure 30a reveals that gene MYD88 seems to act as a local hub for the core cliques. This is confirmed by Figure 30c), where MYD88 is the only genes appearing in all core cliques (except in the isolated one).

For this MHCS all biological situations in which the genes are overexpressed correspond to activities of cells from the immune system, more precisely, they are all overexpressed simultaneously in four situations corresponding to normal activities of white blood cells. Here again, this is very coherent with the L2L list associated to the whole MHCS (p-value of 3.29×10^{-15}) that is entitled *immune response*. The L2L lists which correspond to the different core cliques that compose this MHCS are given Fig. 30c. While one of the core clique is tagged (as the whole MHCS) as being involved in the general immune response process, three other are associated to more specific immune responses: *response to virus* and *inflammatory response*. Finally two other core cliques form a set of genes involve in a regulation process and the last one (the one not directly connected to the other core cliques) is associated to a signal transduction process.

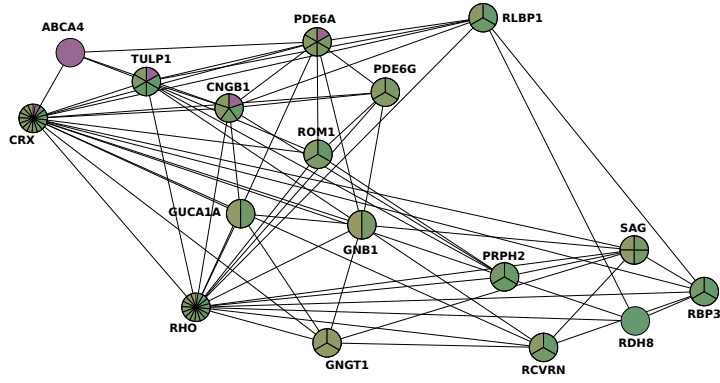
6.2.3.2 Analysis of two CoHoPs

Similarly to what have been done for the MHCSs, we used the results presented Figure 28 to set the parameters, but we also wanted the CoHoPs to share as much biological situations as possible. The parameter setting $\alpha = 4$, $\gamma = 3$ and $k = 3$ provides a collection of 25 CoHoPs with at least 70% of them having a p-value that satisfies to the significance threshold retained (Figure 28, top right graph).

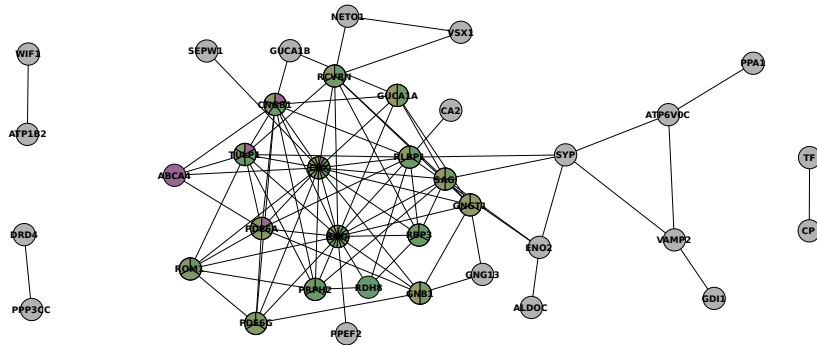
We describe two CoHoPs from this collection.

The first one is presented Figure 31. It is associated to the L2L list “mitochondrial electron transport, NADH to ubiquinone” with a p-value of 9.86×10^{-10} , and describes a group of genes that are simultaneously overexpressed in four biological situations of a specific breast carcinoma cell line (ZR75). Three groups of genes appear in the pattern as three k-PCs. This exhibits three protein modules, all corresponding to different very specific functions, as shown by the three associated L2L lists given Table 31b (and confirmed by the expert), but all being active in the same four biological situations.

The CoHoPs can also contain groups with similar functions, as for instance the second CoHoP presented Figure 32. This CoHoP is associated to the L2L list “intracellular signalling cascade” with a p-value



(a) Only κ -maximal cliques. A colour corresponds to each κ -maximal clique, vertices in multiple colours are part of several 4-maximal cliques.

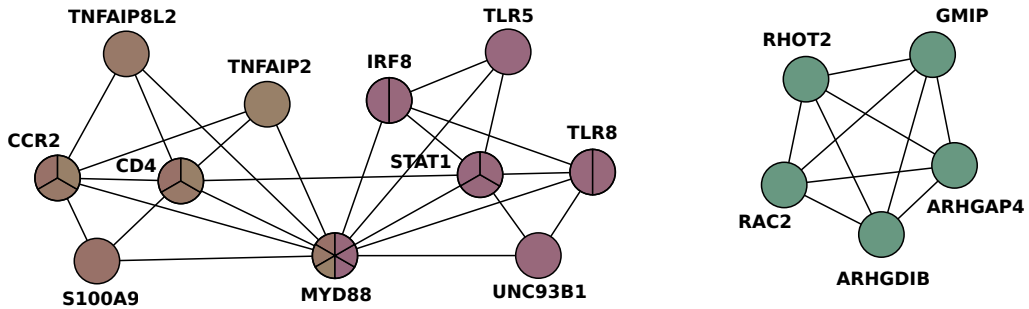


(b) All cliques except single node cliques. A colour corresponds to each κ -maximal clique. Vertices in grey are not in a 4-maximal clique.

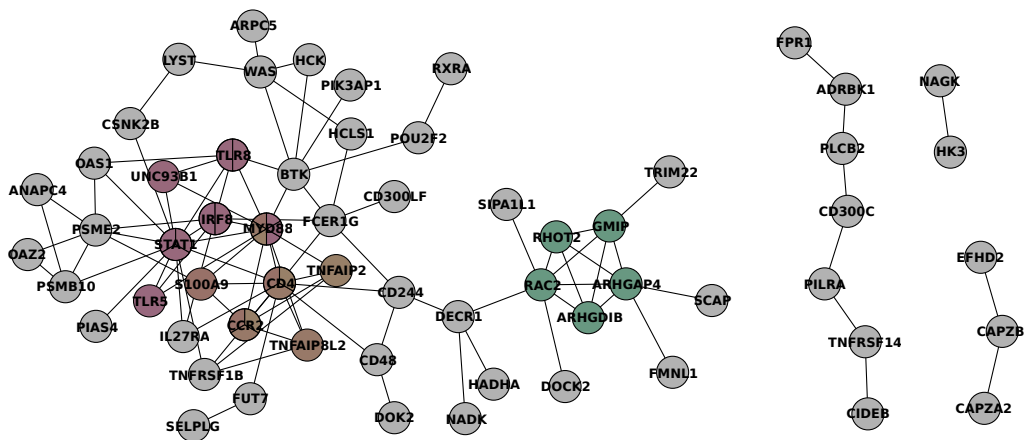
p-value	L2L list	Protein module
9.89×10^{-13}	perception of light	CNGB1,CRX,PDE6A,RHO,RLBP1,TULP1
8.09×10^{-11}	perception of light	CRX,PDE6A,PDE6G,RHO ROM1
8.09×10^{-11}	perception of light	CRX,RBP3,RCVRN,RHO,SAG
8.09×10^{-11}	perception of light	CNGB1,CRX,PRPH2,RHO,TULP1
8.09×10^{-11}	perception of light	ABCA4,CNGB1,CRX,PDE6A,TULP1
8.09×10^{-11}	perception of light	CRX,PRPH2,RHO,ROM1,TULP1
8.09×10^{-11}	perception of light	CRX,PDE6A,RHO,ROM1,TULP1
8.09×10^{-11}	perception of light	CNGB1,CRX,PDE6A,PDE6G,RHO
2.91×10^{-09}	phototransduction	CRX,RCVRN,RHO,TULP1
5.69×10^{-09}	phototransduction	CRX,GNGT1,RCVRN,RHO,SAG
6.71×10^{-09}	perception of light	RBP3,RDH8,RHO,RLBP1
6.71×10^{-09}	perception of light	CNGB1,CRX,GUC1A,RHO
6.71×10^{-09}	perception of light	CRX,PRPH2,RHO,SAG
6.71×10^{-09}	perception of light	CRX,RBP3,RHO,RLBP1
1.63×10^{-08}	perception of light	CRX,GNGT1,GUC1A,RHO,SAG
1.63×10^{-08}	perception of light	CRX,GNB1,PDE6A,PDE6G,RHO
7.36×10^{-05}	sensory perception	CRX,GNB1,GNGT1,RHO

(c) p-value and L2L list for the cliques having at least 4 vertices

Figure 29.: MHCS extracted from dataset BioData₄₀₀ with $\alpha = 3$, $\kappa = 4$ and $\gamma = 3$. All genes are overexpressed in 3 biological situations corresponding to normal activities of retinal cells.



(a) Only 4-maximal cliques. A colour corresponds to each κ -maximal clique, vertices in multiple colours are part of several 4-maximal cliques.

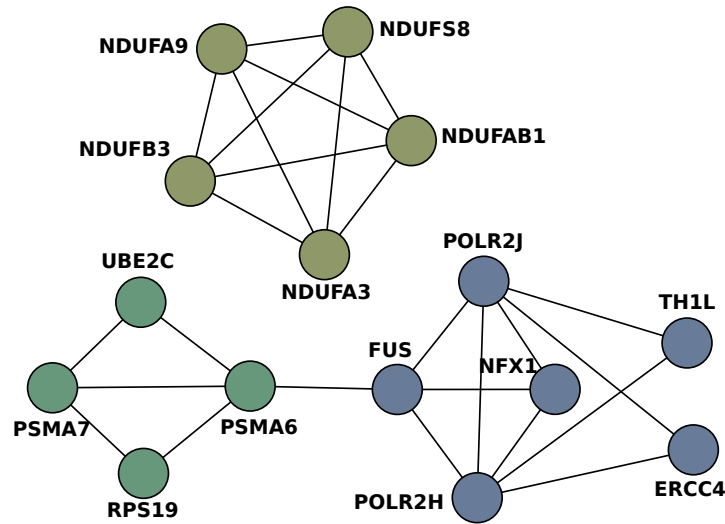


(b) All cliques except single node cliques. A colour corresponds to each κ -maximal cliques. Vertices in grey are not in a 4-maximal cliques.

p-value	L2L list	Protein module
1.22×10^{-07}	response to virus	IRF8,MYD88,STAT1,TLR8
1.22×10^{-07}	response to virus	MYD88,STAT1,TLR8,UNC93B1
4.39×10^{-07}	signal transduction	ARHGAP4,ARHGDIB,GMIP,RAC2,RHOT2
4.15×10^{-06}	inflammatory response	CCR2,CD4,MYD88,S100A9
8.83×10^{-06}	regulation process	CCR2,CD4,MYD88,TNFAIP8L2
8.83×10^{-06}	regulation process	CCR2,CD4,MYD88,TNFAIP2
4.11×10^{-05}	immune response	IRF8,MYD88,STAT1,TLR5

(c) L2L list (middle column) having the best p-value (left column) for each protein module (right column) forming the CoHoP.

Figure 30.: MHCS extracted from dataset BioData₄₀₀ with $\alpha = 3$, $\kappa = 4$ and $\gamma = 3$. All genes are overexpressed in 4 biological situations corresponding to normal activities of white blood cells. In the table, *signal transduction* and *regulation process* are short names for respectively *small GTPase mediated signal transduction* and *positive regulation of cytokine biosynthetic process*.



(a) Protein protein interaction network for 15 proteins forming the three protein modules in the CoHoP. A colour corresponds to each k-PC.

p-value	L2L list	Protein module
4.15×10^{-12}	mitochondrial electron transport, NADH to ubiquinone	NDUFA3,NDUFA9,NDUFAB1 NDUFB3,NDUFS8
1.75×10^{-05}	ubiquitin-dependent protein catabolic process	PSMA6,PSMA7,RPS19,UBE2C
6.45×10^{-03}	negative regulation of transcription	ERCC4,FUS,NFX1,POLR2H,POLR2J TH1L

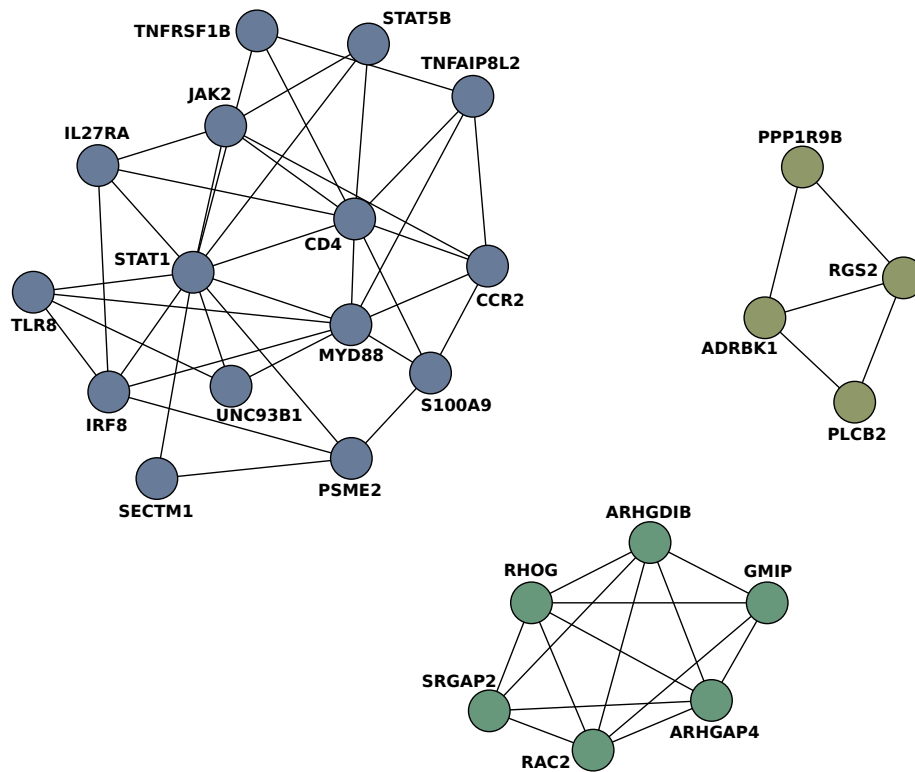
(b) L2L list (middle column) having the best p-value (left column) for each protein module (right column) forming the CoHoP.

Figure 31.: A CoHoP extracted from dataset BioData₄₀₀ with $\alpha = 4$, $k = 3$ and $\gamma = 3$. The genes are overexpressed in four biological situations related to breast carcinoma cell line, ZR75.

of 3.51×10^{-7} . It contains three k-PCs associated to the L2L lists given Table 32b, all corresponding to functions involved in signalling pathways, according to the expert knowledge. We can also notice that if we had considered MHCS instead of CoHoP, the groups formed by four vertices in Figure 31 (UBE2C, PSMA6, RPS19, PSMA7) and Figure 32 (ADRBK1, PPP1R9B, RGS2, PLCB2) would have been split since they are formed by 3-cliques.

6.3 PERFORMANCE EVALUATION

Here we present a quantitative evaluation of the MHCSs and CoHoPs patterns on the biological datasets. The experiments were performed on a PC running GNU/Linux with a 3 GHz Core 2 Duo CPU and 8 GB of main memory installed. Regarding runtime, Figure 33 shows that all extractions can be made in less than 6 seconds and that the runtime is smaller when extracting CoHoP patterns compared to MHCS, except when $\alpha = 1$.

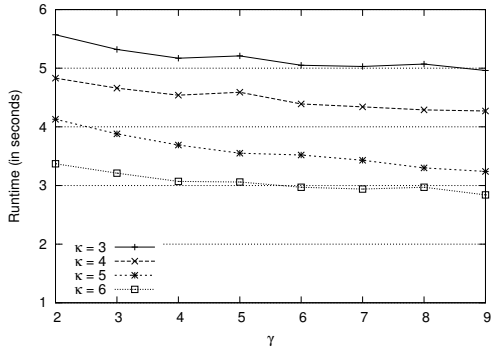


(a) Protein protein interaction network for 15 proteins forming the three protein modules in the CoHoP. A colour corresponds to each k-PC.

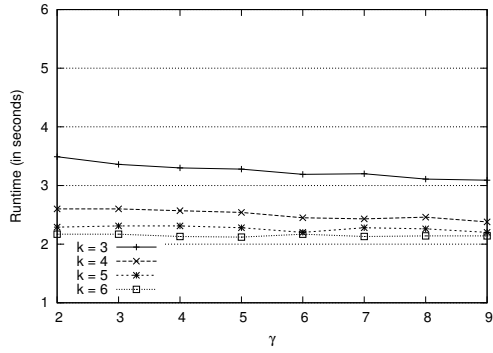
p-value	L2L list	Protein module
1.02×10^{-07}	JAK-STAT cascade	CCR2,CD4,IL27RA,IRF8,JAK2,MYD88,PSME2 S100A9,SECTM1,STAT1,STAT5B,TLR8 TNFAIP8L2,TNFRSF1B,UNC93B1
1.55×10^{-05}	Rho protein signal transduction	ARHGAP4,ARHGDIB,GMIP,RAC2 RHOG,SRGAP2
5.91×10^{-05}	regulation of G-protein	ADRBK1,PLCB2,PPP1R9B,RGS2

(b) L2L list (middle column) having the best p-value (left column) for each protein module (right column) forming the CoHoP.

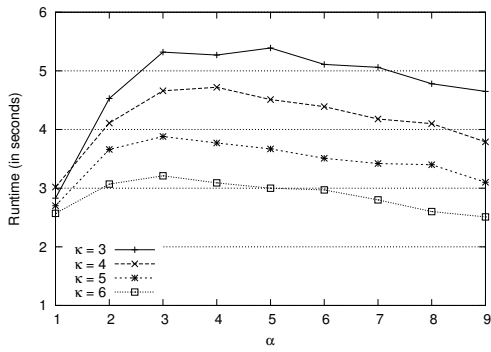
Figure 32.: A CoHoP extracted from dataset BioData₄₀₀ with $\alpha = 4$, $k = 3$ and $\gamma = 3$. The genes are overexpressed four biological situations related to antigen-purified CD14+ monocytes.



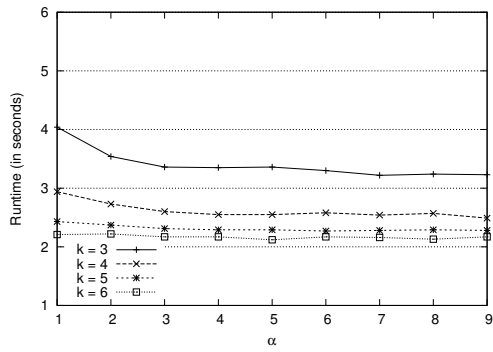
(a) MHCs on BioData₁₅₀ with $\alpha = 3$.



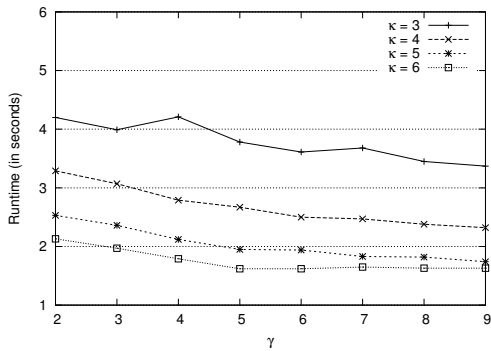
(b) CoHoPs on BioData₁₅₀ with $\alpha = 3$.



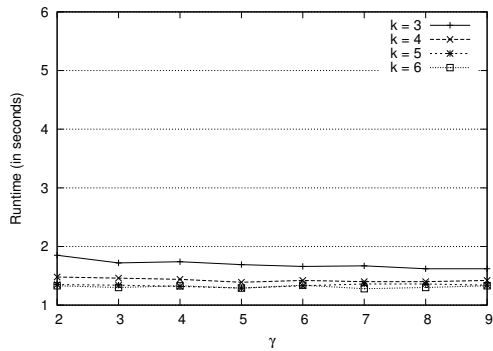
(c) MHCs on BioData₁₅₀ with $\gamma = 3$.



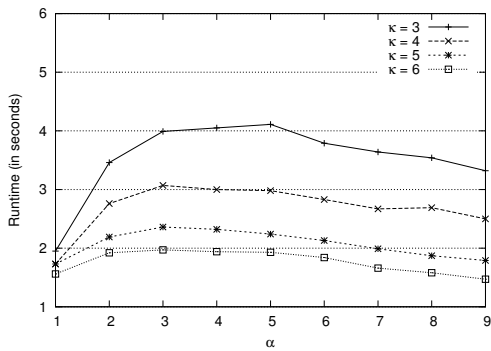
(d) CoHoPs on BioData₁₅₀ with $\gamma = 3$.



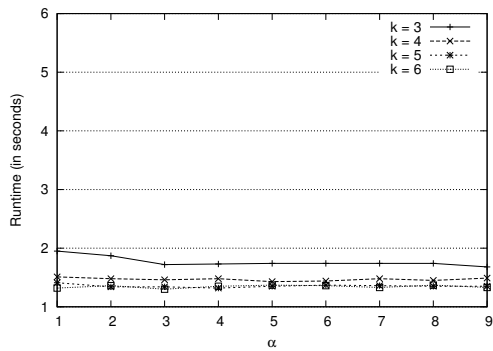
(e) MHCs on BioData₄₀₀ with $\alpha = 3$.



(f) CoHoPs on BioData₄₀₀ with $\alpha = 3$.



(g) MHCs on BioData₄₀₀ with $\gamma = 3$.



(h) CoHoPs on BioData₄₀₀ with $\gamma = 3$.

Figure 33.: Extraction runtime for MHCs and CoHoPs using different sets of parameters on BioData₁₅₀ and BioData₄₀₀.

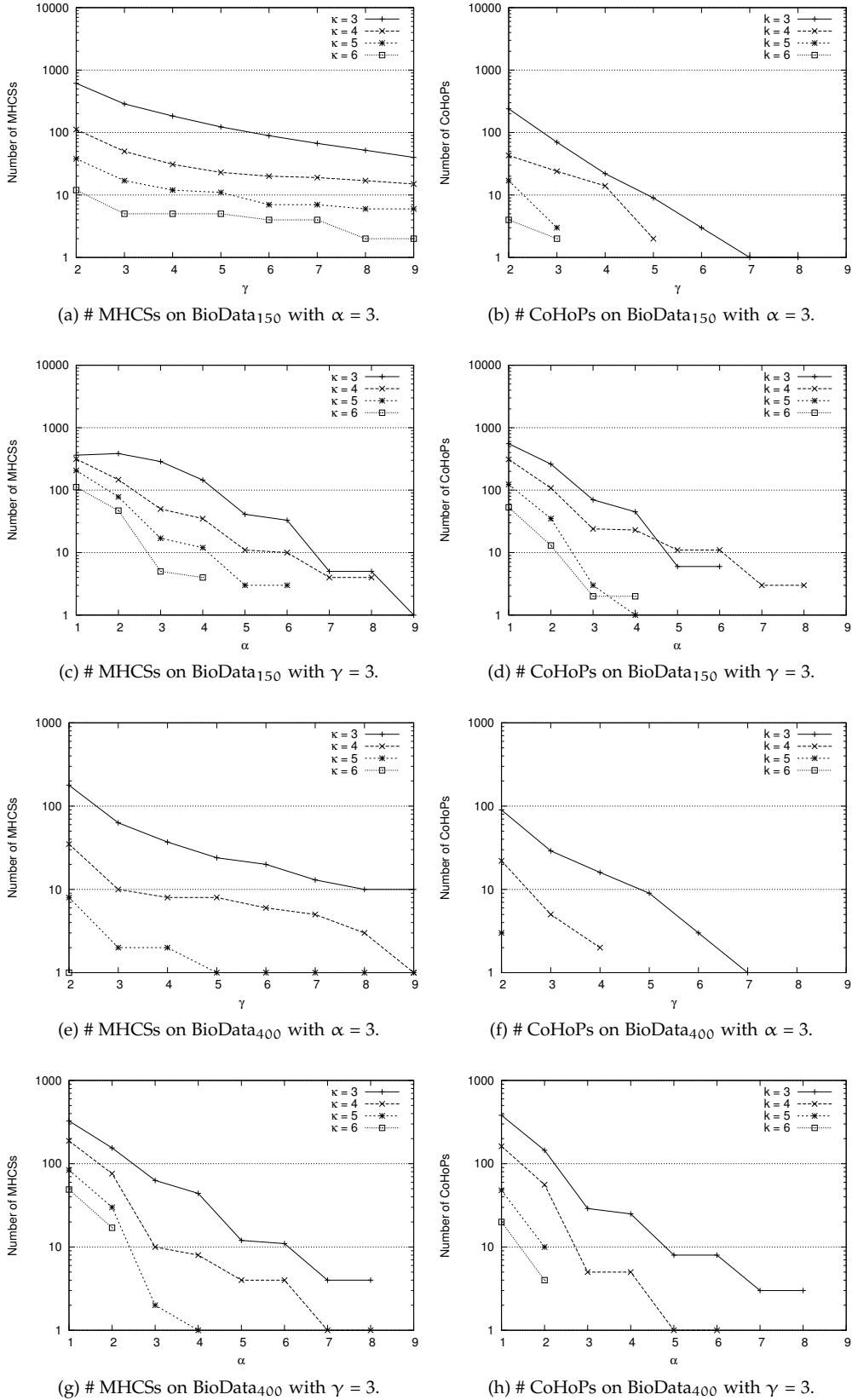


Figure 34.: Number of extracted MHCSs and CoHoPs using different sets of parameters on BioData₁₅₀ and BioData₄₀₀. The line is not drawn if no pattern are output for a set of parameters.

CONCLUSION

In this part, we applied our results in the context of a protein-protein interaction networks with biological situations associated to the proteins. Using a p-value measure, we experimentally shown that for reasonable ranges of parameters, most of the extracted groups are coherent with the current knowledge in biology.

Moreover, in collaboration with a domain expert, we were able to interpret several patterns. We also provided a performance evaluation showing that the MHCSs and CoHoPs can be extracted in a few seconds for real settings.

Due to the data collection process, which might lead to missing values, the CoHoP patterns seems to be well adapted for the discovery of set of groups in these data (a dense subgraph with a few missing edges is less likely to be counted as several groups). Moreover, the presented runtimes indicate that extracting CoHoPs was faster than extracting MHCSs in most experiments on these data. However, the MHCSs remain interesting when one wants to find sets of groups formed by objects being all pairwise connected.

Part V
CONCLUSION

CONCLUSION

In this thesis, we proposed to identify collections of homogeneous groups in attributed graphs. This work complete the existing approaches that consider patterns formed by either (1) a single dense homogeneous group or (2) a set of homogeneous but not necessarily dense groups.

SUMMARY OF OUR CONTRIBUTIONS

Our main contribution is the definition of two classes of patterns used to compute homogeneous collections of dense subgraphs.

We first proposed the Maximal Homogeneous Clique Sets (MHCS), which are formed by collections of cliques where the vertices are homogeneous with respect to the values of the attributes. The second class of patterns, named Collection of Homogeneous k -clique Percolated components (CoHoP) extends the MHCS in order to allow missing values in the pattern. They are based on a relaxed notion of density which allows the subgraphs not to be fully pairwise connected. Two constraint-based algorithms were proposed in order to compute the complete collections of patterns. These algorithms take advantage of several pruning techniques, for which we provided proof of correctness, to reduce the search space. The practical interest of the MHCSs and the CoHoPs are presented in two contexts, a scientific collaboration network and a protein protein interaction network. The scalability and the performances of our algorithms were also studied by performing experiments on synthetic datasets and large real datasets.

We also developed a software tool in order to extract, visualise and browse collections of patterns. This software has been used by another group of researchers for the analysis of large texts. Their results have been published in [72]. They propose to extract CoHoP patterns in an attributed graph where the vertices represent sentences and the attributes associated to a vertex are the lexical units in the corresponding sentence. They show that the CoHoPs can be used to study the structures called *sentence networks* in linguistic.

FUTURE DIRECTIONS OF WORK

Browsing general and specific patterns

A possible improvement for the analysis of CoHoP patterns is to give the ability to browse the collection of more general or more specific patterns. More precisely, given X a CoHoP pattern associated to a set of attributes, another pattern is considered more general if it is associated to a subset of these attributes and more specific if it is associated to a superset of these attributes. The more general patterns give the context of the pattern X with respect to some attributes, while a more specific ones highlights a subpart of X (a subgraph sharing more attributes). Indeed, very preliminary browsing facilities have already been implemented in our visualisation software. An example of results is presented Figure 35.

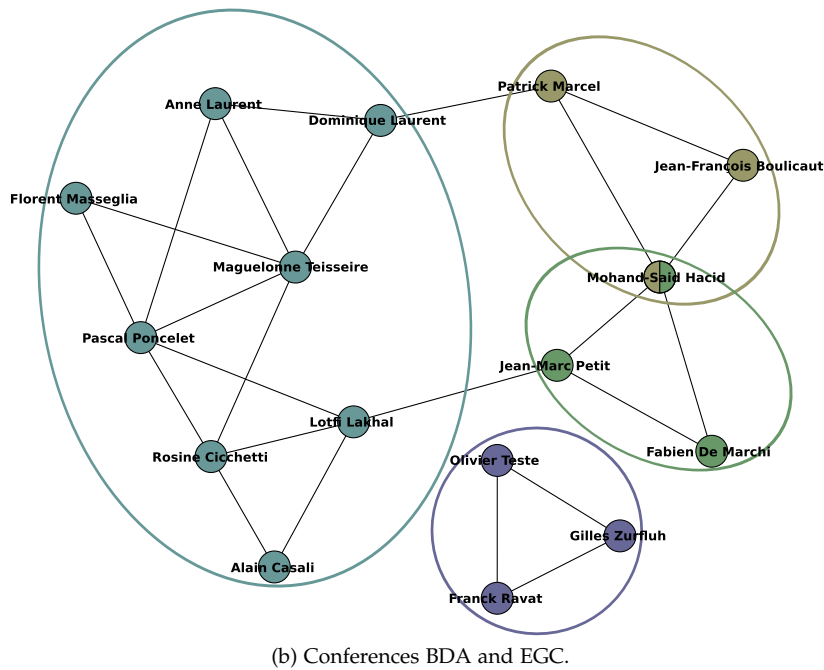
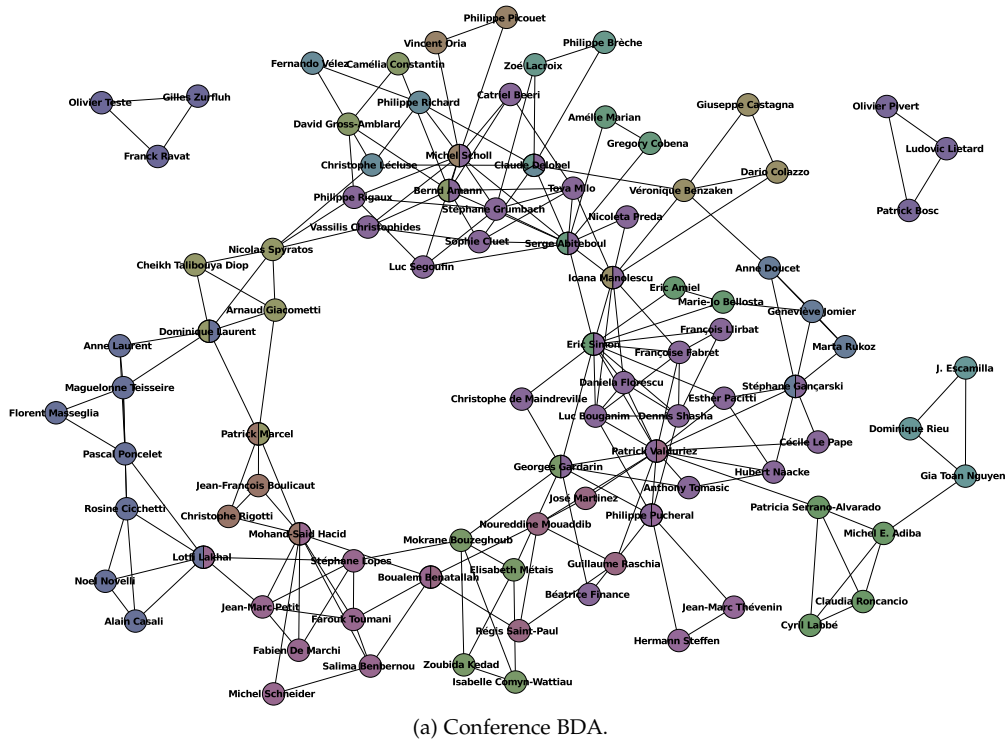


Figure 35.: A general and a more specific CoHoP in a bibliographic dataset.

Tolerating missing values in the attribute space

Another track of improvements would be to allow missing attributes in the set of shared attributes. Indeed, in our pattern definitions, a vertex connected to all vertices of a group in a pattern, but sharing all the attributes of a pattern except one, would not be part of the pattern. Such tolerance would be particularly useful in the context of the application of the MHCS and CoHoP to protein interaction networks presented Section 6. The measurement of gene expression is particularly difficult and can lead to fail to identify with sufficient confidence some situations where a gene is overexpressed. This will result in missing values in the set of situations associated to the corresponding protein in the data collected.

One way to avoid this problem would be to extend the definition of our patterns in order to find structures such as the one presented Figure 36. Using parameters $\alpha = 2$, $\gamma = 2$, and $k = 3$ or $\kappa = 3$, in this example, since attributes a_2 is not shared by vertex E, the homogeneity constraint would not be satisfied for both the MHCS and the CoHoP patterns.

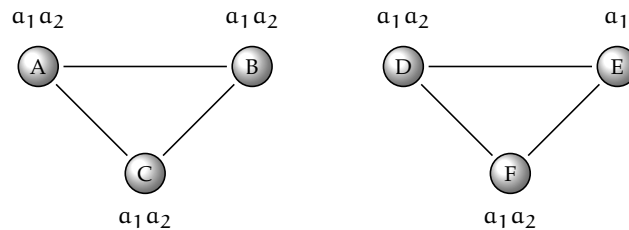


Figure 36.: Example of subgraph where all vertices share two common attributes except vertex E.

Extension to other data structures

As mentioned in the introduction of this thesis, a challenge in data mining is the ability to handle complex data. Toward this aim, we could extend our approach to handle Boolean attributes, not only on the vertices, but also on the edges of the graphs. For instance, this could be used to encode multidimensional networks [8] (a multidimensional network is a network containing different kinds of relationships between its nodes). For example, consider the multidimensional network depicted Figure 37 with attributes associated to vertices (rock, pop, and jazz), and where the edges represent three different kinds of relationships: friendship, geographic closeness or being colleague at work. This network can be encoded as the attributed graph, with Boolean attributes associated to the edges, given Figure 38. Then, an extension of the MHCS and CoHoP patterns to handle these data could help to identify structures in groups sharing attributes on the vertices and on the edges like the one shown Figure 39.

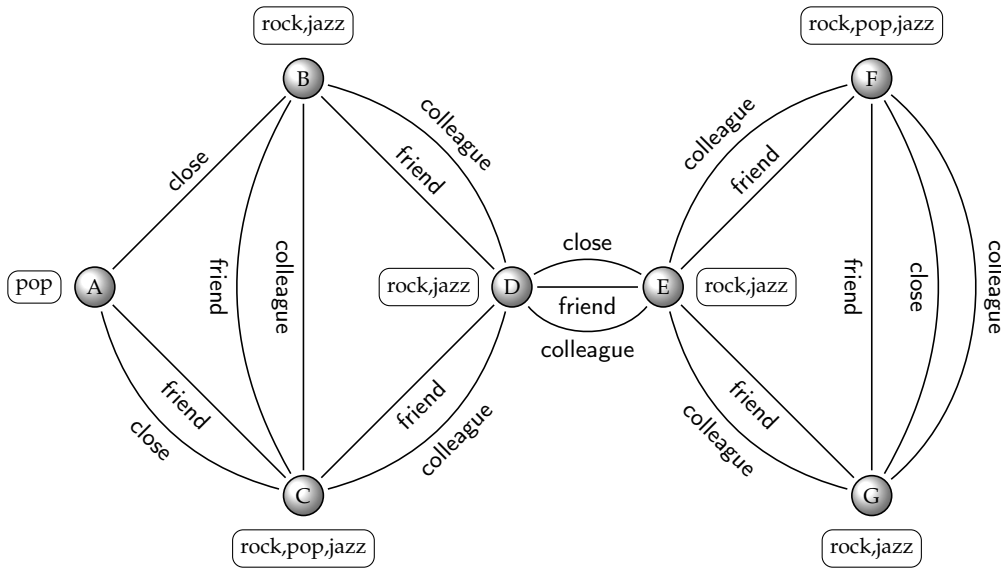


Figure 37.: Example of multidimensional network with attributes associated to the nodes.

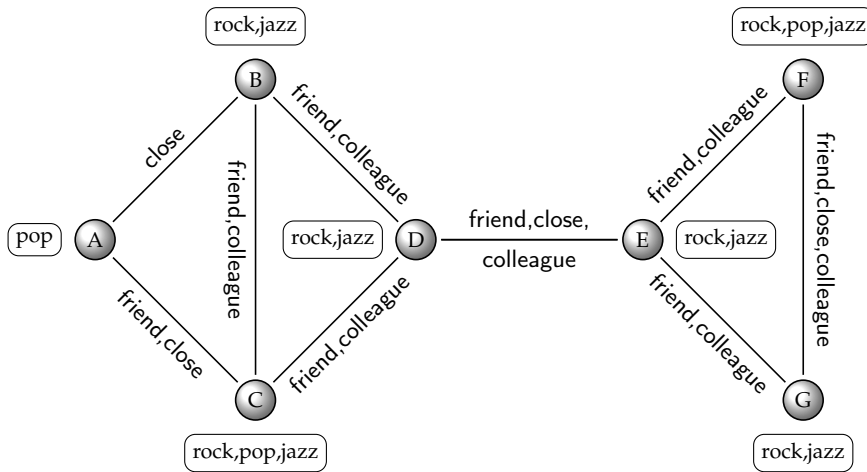


Figure 38.: The multidimensional network presented Figure 37 where edge labels are encoded by means of Boolean attributes in an attributed graph.

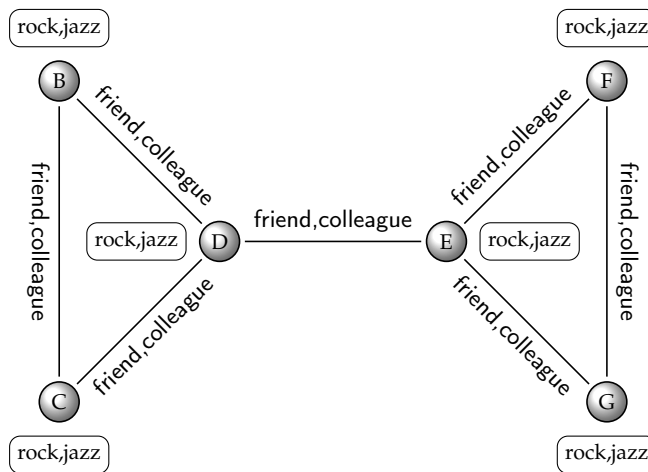


Figure 39.: Example of structure which might be found in the graph presented Figure 38. It is formed by two groups of three vertices sharing two vertex attributes (rock and jazz) and two edge attributes (friend and colleague).

APPENDIX

DATASET DESCRIPTION

In this appendix, we provide a description of the dataset properties. Most results were obtained using the Gephi software [4] available at <https://gephi.org/>.

A.1 THE BIODATA DATASETS

We first present the graph characteristics for the datasets used in the application to molecular biology. Note that due to the dataset construction method, only the number of edges is different between BioData₁₅₀ and BioData₄₀₀. Table 8 summarizes several graph characteristics for both datasets. Figure 40 displays the largest connected component of BioData₄₀₀. The other components are not displayed since most of them are much smaller and reduce graph understanding. Figure 41 presents the distribution of the degree and the distribution of the number of attribute having value True for BioData₁₅₀ and BioData₄₀₀.

Measure	BioData ₁₅₀	BioData ₄₀₀
# Vertices	15,571	15,571
# Attributes	486	486
Avg. degree	58.92	20.01
Max. degree	1500	496
Avg. attributes/vertex	11.46	11.46
Max. attributes/vertex	92	92
# Connected Components (CC)	6,288	5,759
# Vertices in the largest CC	9805	9267
% Vertices in the largest CC	63%	59.5%
Number of maximal cliques	-	122,186
# Vertices in the largest clique	-	139
Average clustering coefficient	0.19	0.15
# Triangles	6,267,990	1,081,691
Modularity [9]	-	0.474
Diameter	-	8

Table 8.: Main characteristics of the datasets BioData₁₅₀ and BioData₄₀₀. Dashes denote prohibitive runtime or memory consumption.

A.2 THE DBLP DATASETS

Graph characteristics corresponding to the bibliographic dataset are presented Figure 9. Figure 42 displays the largest connected component of DBLP₃. The other components are much smaller and are not displayed to ease graph visualisation. Figure 43 presents the distribution

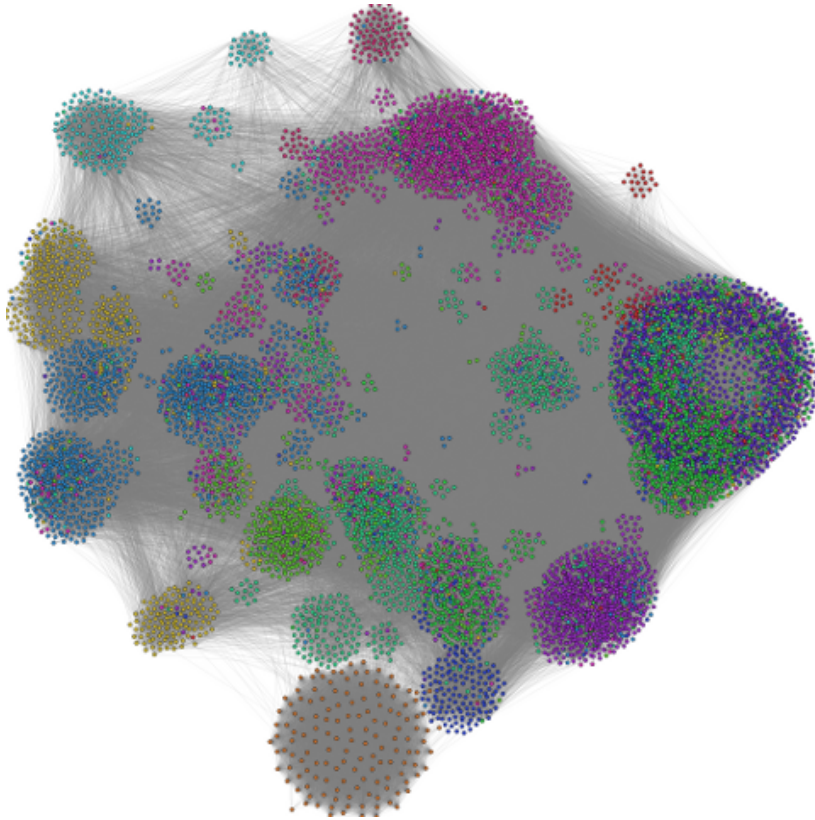


Figure 40.: The largest connected component of BioData_{400} . Each colour corresponds to a module according to the modularity definition of [9].

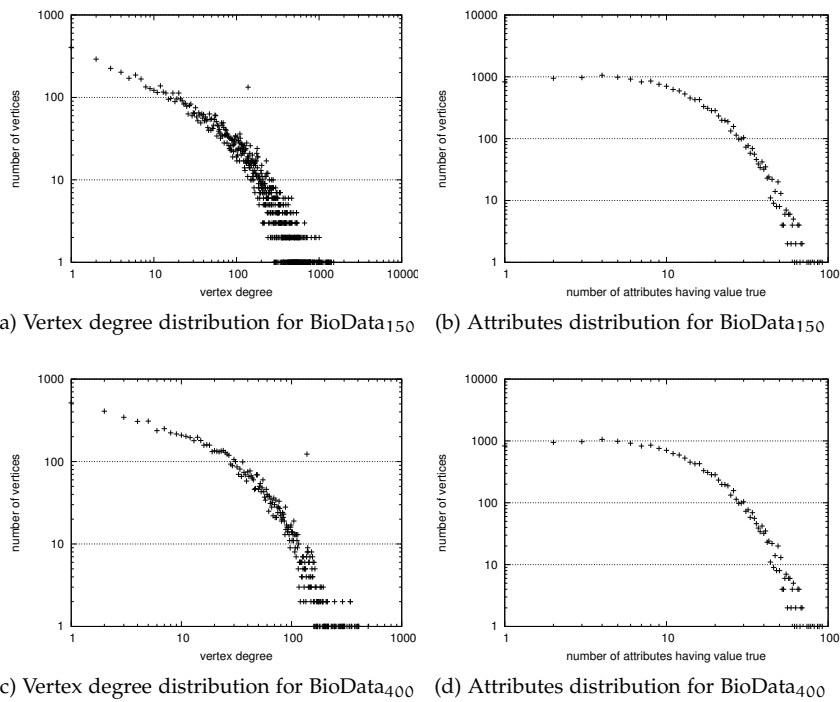


Figure 41.: Degree and attribute distribution for the datasets BioData_{150} and BioData_{400} . All the scales are logarithmic.

of the degree and the distribution of the number of attribute having value True for $DBLP_1$, $DBLP_2$ and $DBLP_3$.

Measure	$DBLP_1$	$DBLP_2$	$DBLP_3$
# Vertices	997,050	266,125	127,386
# Attributes	5,963	5,309	3,980
# Edges	3,427,683	650,205	234,896
Avg. degree	6.88	4.89	3.69
Max. degree	1,014	240	149
Avg. attributes/vertex	3.06	2.44	2.15
Max. attributes/vertex	302	112	68
# Connected Components (CC)	93,644	44,047	28,236
# Vertices in the largest CC	817,717	198,980	86,643
% Vertices in the largest CC	82%	74.8%	68%
Number of maximal cliques	-	273,810	128,572
# Vertices in the largest clique	-	55	45
Average clustering coefficient	0.6	0.41	0.32
# Triangles	7,250,116	773,379	213,285
Modularity [9]	-	0.78	0.818
Diameter	-	27	28

Table 9.: Main characteristics of the datasets $DBLP_1$, $DBLP_2$, and $DBLP_3$. Dashes denote prohibitive runtime or memory consumption.

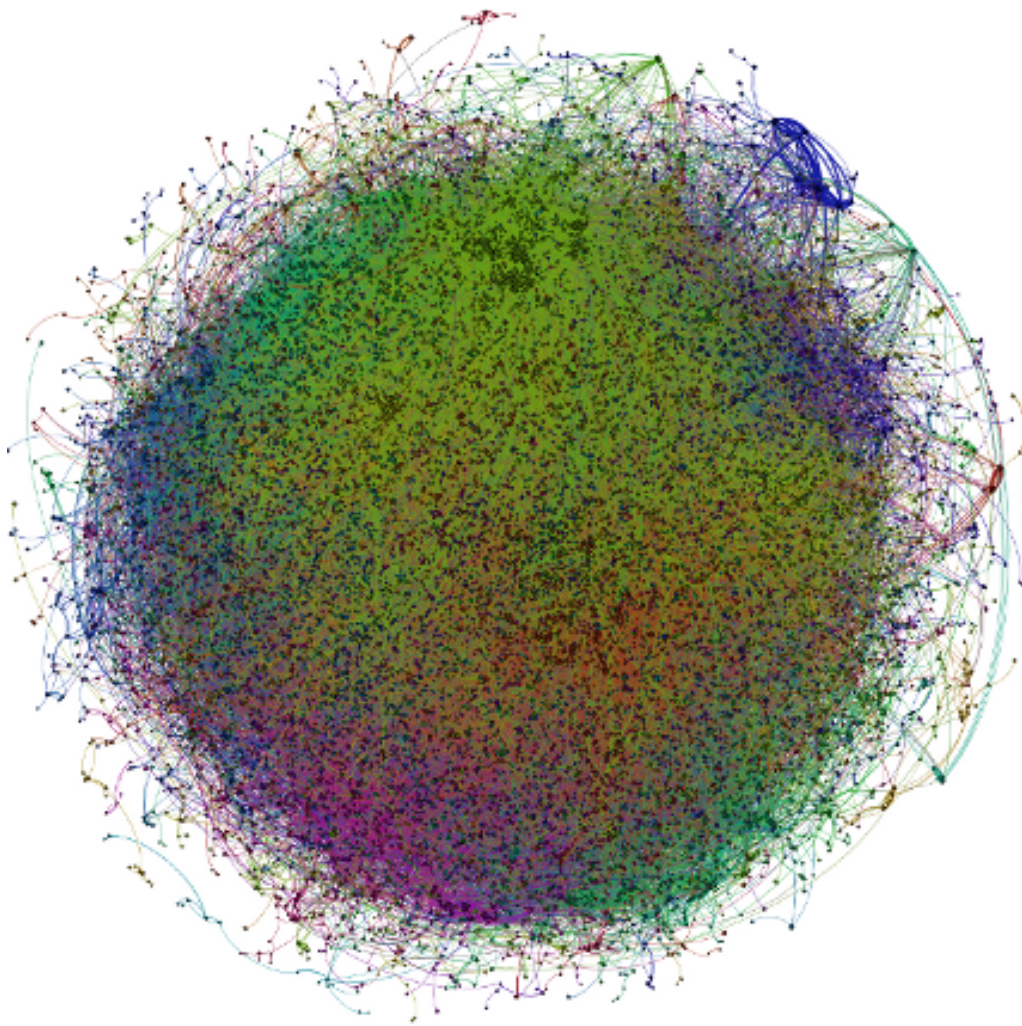


Figure 42.: The largest connected component of $DBLP_3$. Each colour corresponds to a module according to the modularity definition of [9].

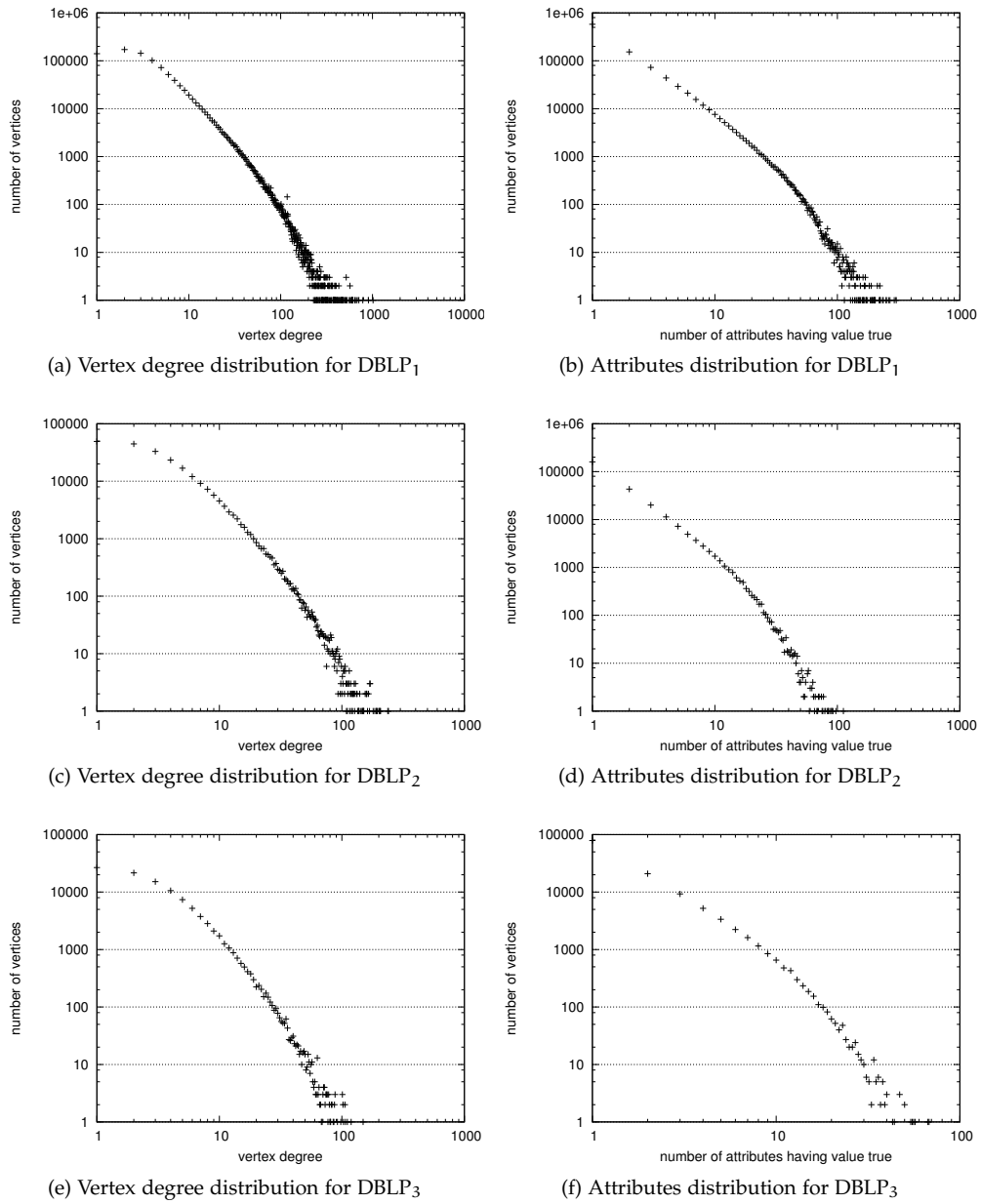


Figure 43.: Degree and attribute distribution for the datasets DBLP₁, DBLP₂, and DBLP₃. All the scales are logarithmic.

A PATTERN EXTRACTION / VISUALISATION SOFTWARE

In order to ease the extraction, the browsing and the visualisation of the collections of patterns, we developed a dedicated software tool. It has been implemented using the Java 1.6 language.

B.1 PRESENTATION OF THE INTERFACE

The interface is composed of a taskbar (at the top in Figure 44) allowing to set extraction parameters and the three following views:

- Pattern visualisation (at the left in Figure 44);
- The collection of patterns (at the bottom right in Figure 44);
- Vertex measures (at the top right in Figure 44);

The view layout is highly customizable since it is based on the library Docking Framework¹. Docking allows for example to resize and move views or even group views using tabs.

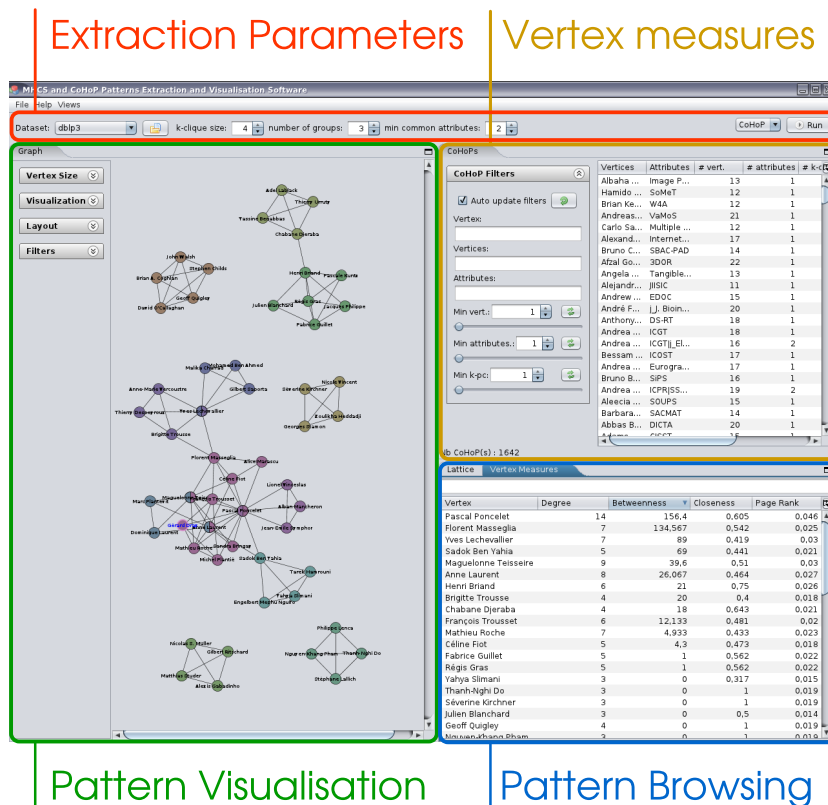


Figure 44.: The different parts of the interface

In the next sections, the parts of the interface are described, along with their functionalities.

1. The Docking Framework library is available at <http://dock.javaforge.com/>.

B.2 EXTRACTION OF THE PATTERNS

The first step to use the software is to perform a pattern extraction (either MHCS or CoHoP patterns). The taskbar at the top of the window allows to set the extraction parameters. The user has to define:

- the dataset of interest;
- the value for parameter k (CoHoP) or κ (MHCS) (“size” in the interface);
- the value for parameter γ (“number of groups” in the interface).
- the value for parameter α (“min common attributes” in the interface).
- the task, either MHCSs or CoHoPs extraction.

Once all parameters are defined, the extraction can be performed by clicking the “Run” button at the right side of the taskbar. In fact, for a given set of parameters the extraction will be performed only the first time. Once the extraction has finished, the collection of patterns is saved into a file. Further extractions using the same settings will use this file instead of recomputing the collection of patterns from the data.

B.3 BROWSING THE COLLECTION OF PATTERNS

Once the collection of patterns is computed, it is displayed on the view presented Figure 45 using a tabular form. Each row corresponds to a pattern. A row contains the following information:

- names of the vertices forming the pattern;
- names of the attributes associated to the vertices;
- number of vertices;
- number of attributes;
- number of k -PCs for the CoHoP patterns or k -maximal cliques for the MHCS patterns.

It is possible to sort the collection of patterns according to any of these values by clicking on their corresponding column header.

Vertices	Attributes	# vert.	# attributes	# k-clique	# sub-patt.	# sup-patt.
Erik D. Demaine,Jack Snoeyink,...	CCCG _Discrete...	12	6	3	1	34
Erik D. Demaine,Günter Rote,Ja...	CCCG _Comput...	12	6	3	1	40
Erik D. Demaine,Günter Rote,Ja...	CCCG _Comput...	13	5	3	2	22
Erik D. Demaine,Ferran Hurtado...	CCCG _Discrete...	14	6	4	2	43
Erik D. Demaine,Ferran Hurtado...	CCCG _Discrete...	15	5	5	5	21
Anna Lubiw,Erik D. Demaine,Her...	CCCG _Comput...	16	5	4	2	18
David R. Wood,Erik D. Demaine,...	CCCG _Discrete...	16	5	4	4	22
David R. Wood,Erik D. Demaine,...	CCCG _Discrete...	17	4	5	9	10
Erik D. Demaine,Esther M. Arkin...	CCCG _Comput...	18	5	3	4	23
Anil Maheshwari,Anna Lubiw,Er...	WADS CCCG _C...	18	4	3	6	12
Anna Lubiw,Erik D. Demaine,Est...	SODA CCCG _Co...	18	5	4	2	21
Erik D. Demaine,Esther M. Arkin...	CCCG _Comput...	19	4	4	9	11
Anna Lubiw,Erik D. Demaine,Est...	SODA CCCG _Co...	19	4	4	4	11
David Bremner,Erik D. Demaine,...	CCCG _Discrete...	22	5	4	5	24
David Bremner,David R. Wood,E...	CCCG _Discrete...	24	4	4	9	12
Anil Maheshwari,David R. Wood,...	CCCG _Algorith...	25	4	4	11	12
Anil Maheshwari,Anna Lubiw,Bet...	WADS CCCG _C...	26	3	3	10	5
Anil Maheshwari,David R. Wood,...	CCCG _Algorith...	27	3	5	20	5
David Bremner,Diane L. Souvain...	CCCG _Discrete...	28	4	6	11	11
Anna Lubiw,David Bremner,Erik...	CCCG _Comput...	30	4	4	11	12
David Bremner,David R. Wood,Ol...	CCCG _Discrete...	30	3	6	20	5
Anna Lubiw,Bettina Speckmann,...	CCCG _Comput...	39	3	5	22	5
Anil Maheshwari,Anna Lubiw,Da...	CCCG _Comput...	42	3	4	24	6

Figure 45.: The collection of patterns view.

This view allows to filter the collection patterns. The available filters allow to find patterns containing a single vertex, several vertices, or a given attribute. It also allows to keep only patterns having a minimal number of vertices, attributes, maximal cliques for a MHCS, or k -PCs for a CoHoP.

B.4 PATTERN VISUALISATION

Clicking on the row corresponding to a pattern will display the subgraph induced by the vertices forming the pattern in the graph view. Graph visualisation is based on the Jung library². This library allows to easily zoom in/out, translate, and rotate a graph. More actions are available to customize graph appearance using the panels presented Figure 46.

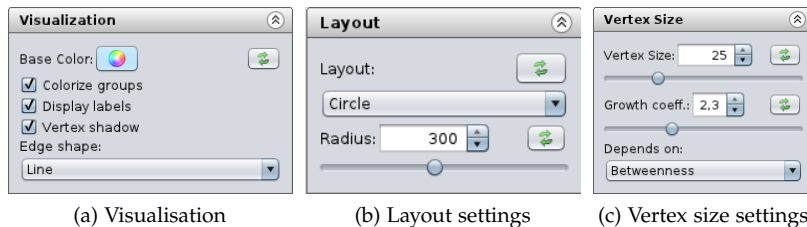


Figure 46.: Panels used to customize pattern visualisation.

B.4.1 Centrality measures

The panel presented Figure 46c allows to easily set the visualisation of several vertex centrality measures (*e.g.*, degree, betweenness). The higher is the measure, the larger the vertex radius will be. For more information on vertex centrality measures refer to the Section 2.1 of the state of the art.

B.4.2 Visualisation parameters

Several graphical options are also available using the panel presented Figure 46a. The user can, for example, change the colour of the vertices, display or not the name of the vertices or change the shape used for the edges. She/he can also give a different colour to each maximal clique (MHCS) or k-PC (CoHoP).

B.4.3 Layout

A common problem for graph visualisation is choosing the right layout. To give the most liberty to the user, she/he can select a layout and edit its parameters. The available layouts are:

- Spring;
- Force directed (Fruchterman-Reingold);
- Circular layout;
- Self-Organized Map;
- Kamada Kawai.

Figure 47 gives an overview of these layout by presenting a CoHoP pattern using each layout algorithm. The reader interested in more information regarding these layouts can refer to the Jung library documentation (<http://jung.sourceforge.net/doc/api/index.html>).

2. <http://jung.sourceforge.net/>

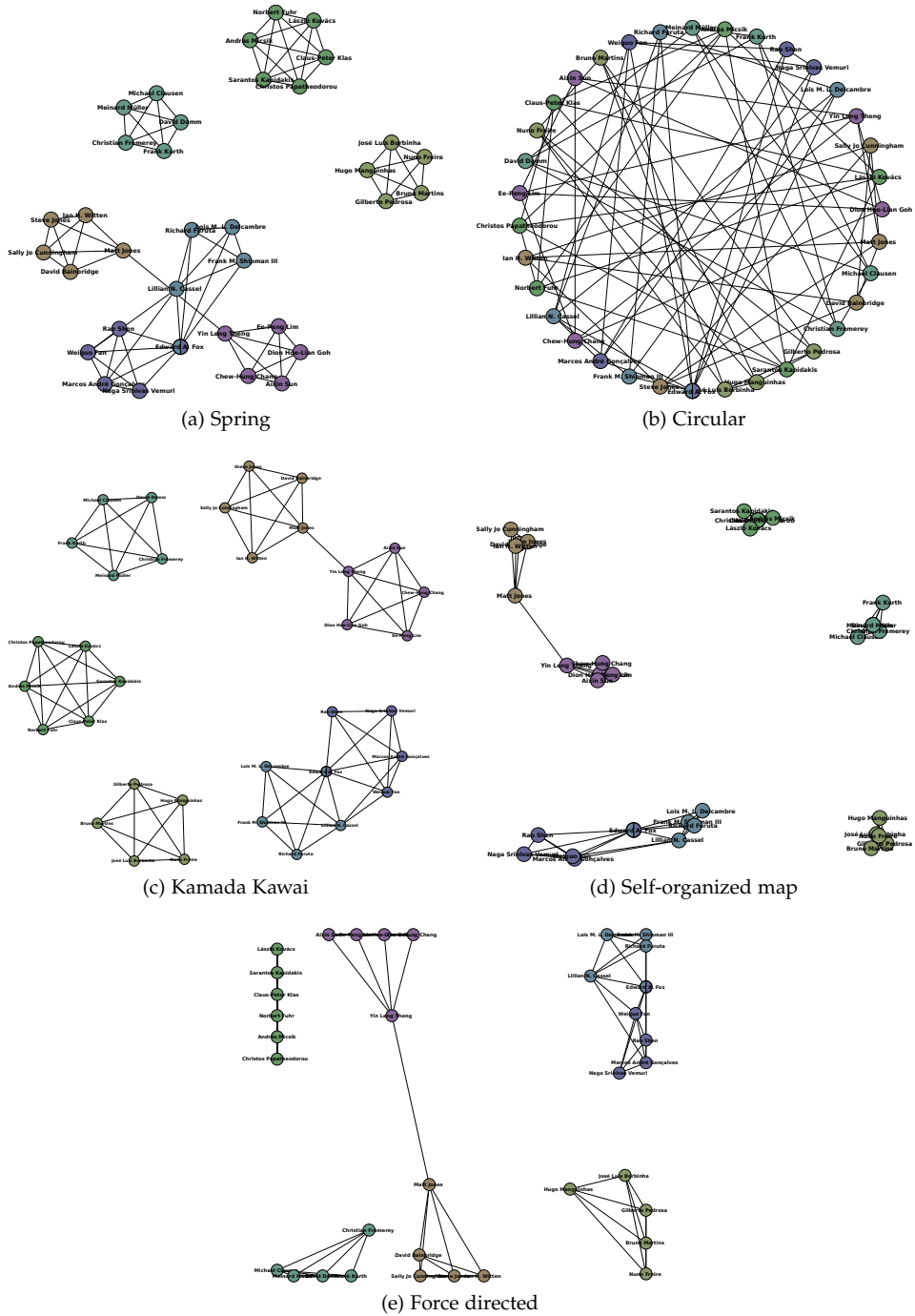


Figure 47.: A CoHoP pattern displayed using the five available layouts.

TABLE OF SYMBOLS

C.1 NOTATION IN THE BINARY RELATION SETTING

Symbol	Description
\mathcal{O}	An arbitrary set of objects
\mathcal{J}	An arbitrary set of items
$\mathcal{R} \subseteq \mathcal{O} \times \mathcal{J}$	A binary relation defined over $\mathcal{O} \times \mathcal{J}$
$f : 2^{\mathcal{O}} \mapsto 2^{\mathcal{J}}$	The set of items associated to all the objects $f(\mathcal{O}) = \{i \in \mathcal{J} \mid \forall o \in \mathcal{O}, (o, i) \in \mathcal{R}\}$
$g : 2^{\mathcal{J}} \mapsto 2^{\mathcal{O}}$	The set of objects associated to all the items $g(\mathcal{I}) = \{o \in \mathcal{O} \mid \forall i \in \mathcal{I}, (o, i) \in \mathcal{R}\}$
$\text{supp} : 2^{\mathcal{J}} \mapsto \mathbb{R}$	The number of objects related to all the items $\text{supp}(\mathcal{I}) = g(\mathcal{I}) $
itemset	A subset of \mathcal{J}
k-itemset	An itemset of size k

Table 10.: Summary of the notation used in the binary relation setting

C.2 NOTATION IN THE GRAPH SETTING

Symbol	Description
\mathcal{V}	An arbitrary set of vertices
\mathcal{E}	An arbitrary set of edges
\mathcal{G}	A graph formed by vertices \mathcal{V} and edges \mathcal{E}
$\mathcal{G}[\mathcal{V}]$	The subgraph induced by the set of vertices \mathcal{V}
$\mathcal{C}(\mathcal{G})$	The collection of cliques in \mathcal{G}
$\mathcal{C}_k(\mathcal{G})$	The collection of cliques having exactly k vertices in \mathcal{G}
$\mathcal{C}_{\max}(\mathcal{G})$	The collection of maximal cliques in \mathcal{G}
$\mathcal{C}_{k\max}(\mathcal{G})$	The collection of maximal cliques having at least k vertices in \mathcal{G}
$\rho(\mathcal{G})$	The density of \mathcal{G} $\rho(\mathcal{G}) = \frac{ \mathcal{E} }{ \mathcal{V} \cdot (\mathcal{V} - 1) / 2}$
$\Gamma(v)$	The neighbours of vertex v $\Gamma(v) = \{v' \in \mathcal{V} \mid \{v, v'\} \in \mathcal{E}\}, v \in \mathcal{V}$

Table 11.: Summary of the notation used in the graph setting

C.3 NOTATION IN THE BOOLEAN ATTRIBUTED GRAPH SETTING

Symbol	Description
\mathcal{V}	An arbitrary set of vertices
\mathcal{E}	An arbitrary set of edges
\mathcal{A}	A set of Boolean attributes
$\text{ATB} : \mathcal{V} \mapsto 2^{\mathcal{A}}$	The set of attributes having value True for the given vertex
\mathcal{G}	A Boolean attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \text{ATB})$
$\text{ATB} : 2^{\mathcal{V}} \mapsto 2^{\mathcal{A}}$	The attributes having value True for all given vertices
$\text{CATB} : 2^{2^{\mathcal{V}}} \mapsto 2^{\mathcal{A}}$	The attributes having value True for all the vertices in the union of the sets given
$\text{VERT} : \mathcal{A} \mapsto 2^{\mathcal{V}}$	The set of vertices having True for the attribute
$\text{VERT} : 2^{\mathcal{A}} \mapsto 2^{\mathcal{V}}$	The set of vertices having True for all the given attributes
$\mathcal{G}[\mathcal{V}]$	The subgraph induced by the set of vertices
$\mathcal{G}[\mathcal{A}]$	The subgraph induced by the set of attributes $\mathcal{G}[\mathcal{A}] = \mathcal{G}[\{v \in \mathcal{V} \mid \mathcal{A} \subseteq \text{ATB}(v)\}]$
$\mathcal{C}(\mathcal{G})$	The collection of cliques in \mathcal{G}
$\mathcal{C}_k(\mathcal{G})$	The collection of k-cliques in \mathcal{G}
$\mathcal{C}_{\text{max}}(\mathcal{G})$	The collection of maximal cliques in \mathcal{G}
$\mathcal{C}_{k\text{max}}(\mathcal{G})$	The collection of k-maximal cliques in \mathcal{G}

Table 12.: Summary of the notation used in the Boolean attributed graph setting

Part VI

BIBLIOGRAPHY

- [1] James Abello, Mauricio G C Resende, and Sandra Sudarsky. Massive Quasi-Clique Detection. In *Proc. of Latin American Symposium on Theoretical Informatics*, pages 598–612. Springer-Verlag, 2002. (Cited on page 23.)
- [2] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pages 487–499, 1994. (Cited on page 14.)
- [3] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of Int. Conf. on Management of Data (SIGMOD)*, pages 207–216. ACM, 1993. (Cited on page 14.)
- [4] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi : An Open Source Software for Exploring and Manipulating Networks. In *Proc. of Int. AAAI Conf. on Weblogs and Social Media*, pages 361–362. The AAAI Press, 2009. (Cited on page 109.)
- [5] Yves Bastide, Rafik Taouil, Nicolas Pasquier, Gerd Stumme, and Lotfi Lakhal. Mining Frequent Patterns with Counting Inference. *SIGKDD Explor. Newsl.*, 2(2):66–75, 2000. (Cited on page 16.)
- [6] Céline Becquet, Sylvain Blachon, Baptiste Jeudy, Jean-François Boulicaut, and Olivier Gandrillon. Strong-association-rule mining for large-scale gene-expression data analysis: a case study on human SAGE data. *Genome Biology*, 3(12):1–16, 2002. (Cited on page 85.)
- [7] Nesserine Benchettara, Rushed Kanawati, and Céline Rouveirol. A Supervised Machine Learning Link Prediction Approach for Academic Collaboration Recommendation. In *Proc. of ACM Conf. on Recommender systems (RecSys)*, pages 253–256. ACM, 2010. (Cited on page 57.)
- [8] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. Foundations of Multidimensional Network Analysis. In *Proc. of Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 485–489. IEEE Computer Society, 2011. (Cited on pages 4 and 105.)
- [9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Statistical Mechanics: Theory and Experiment*, 2008(10): P10008, 2008. (Cited on pages xv, 109, 110, 111, and 112.)
- [10] Francesco Bonchi and Claudio Lucchese. Pushing Tougher Constraints in Frequent Pattern Mining. In *Proc. of European Conf. on Machine Learning and Princ. and Pract. of Knowledge Discovery in Databases (ECML/PKDD)*, pages 114–124. Springer-Verlag, 2005. (Cited on page 18.)
- [11] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi. Exante: Anticipated data reduction in constrained pattern mining. In *Proc. of European Conf. on Machine Learning and Princ. and Pract. of Knowledge Discovery in Databases (ECML/PKDD)*, pages 59–70. Springer-Verlag, 2003. (Cited on page 18.)

- [12] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi. Efficient breadth-first mining of frequent pattern with monotone constraints. *Knowledge and Information Systems (KAIS)*, 8(2):131–153, 2005. (Cited on page 18.)
- [13] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-Sets: A Condensed Representation of Boolean Data for the Approximation of Frequency Queries. *Data Mining and Knowledge Discovery (DMKD)*, 7(1):5–22, 2003. (Cited on page 16.)
- [14] Björn Bringmann and Albrecht Zimmermann. One in a million: picking the right patterns. *Knowledge and Information Systems (KAIS)*, 18(1):61–81, 2009. (Cited on page 43.)
- [15] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, pages 443–452. IEEE Computer Society, 2001. (Cited on page 16.)
- [16] Toon Calders and Bart Goethals. Mining All Non-Derivable Frequent Itemsets. In *Proc. of European Conf. on Machine Learning and Princ. and Pract. of Knowledge Discovery in Databases (ECML/PKDD)*, pages 74–85. Springer-Verlag, 2002. (Cited on page 16.)
- [17] Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Data-Peeler: Constraint-Based Closed Pattern Mining in n-ary Relations. In *Proc. of SIAM Int. Conf. on Data Mining (SDM)*, pages 37–48. SIAM, 2008. (Cited on page 18.)
- [18] Hong Cheng, Philip S Yu, and Jiawei Han. Approximate Frequent Itemset Mining In the Presence of Random Noise. In *Soft Computing for Knowledge Discovery and Data Mining*, pages 363–389. Springer-Verlag, 2008. (Cited on page 5.)
- [19] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A Classification for Community Discovery Methods in Complex Networks. *Statistical Analysis and Data Mining (SADM)*, 4(5):512–546, 2011. (Cited on page 37.)
- [20] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical Review Letters*, 94(16):2–5, 2005. (Cited on page 23.)
- [21] Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu. Community Detection in Large-Scale Social Networks. In *Proc. of WebKDD and SNA-KDD workshop on Web mining and social network analysis*, pages 16–25. ACM, 2007. (Cited on page 30.)
- [22] Paul Erdős and Alfréd Rényi. On Random Graphs. *Publicationes Mathematicae*, 6:290–297, 1959. (Cited on page 60.)
- [23] Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935. (Cited on page 21.)
- [24] 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. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):1–35, 2008. (Cited on page 28.)

- [25] Leonard Euler. *Solutio problematis ad geometriam situs pertinentis*. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741. (Cited on page 19.)
- [26] Linton C Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40(1):35–41, 1977. (Cited on page 20.)
- [27] Mutsumi Fukuzaki, Mio Seki, Hisashi Kashima, and Jun Sese. Finding Itemset-Sharing Patterns in a Large Itemset-Associated Graph. In *Proc. of Pacific-Asia Conf. on Knowl. Discov. and Data Mining (PAKDD)*, volume 1, pages 147–159. Springer-Verlag, 2010. (Cited on pages 31 and 37.)
- [28] Bernhard Ganter, Gerd Stumme, and Rudolf Wille. *Formal Concept Analysis: Foundations and Applications*. Springer-Verlag, 2005. (Cited on page 16.)
- [29] Wei Gao, Kam-Fai Wong, Yunqing Xia, and Ruifeng Xu. Clique Percolation Method for Finding Naturally Cohesive and Overlapping Document Clusters. In *Proc. of Int. Conf. on Computer Processing of Oriental Languages (ICCPOL)*, volume 4285, pages 97–108. Springer-Verlag, 2006. (Cited on pages 23, 69, and 70.)
- [30] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In *Proc. of Discovery Science (DS)*, pages 278–289. Springer-Verlag, 2004. (Cited on page 18.)
- [31] Alain Gély, Lhouari Nourine, and Bachir Sadi. Enumeration aspects of maximal cliques and bicliques. *Discrete Applied Mathematics*, 157(7):1447–1459, 2009. (Cited on page 21.)
- [32] Alan Gibbons. *Algorithmic graph theory*. Cambridge University Press, 1982. (Cited on page 19.)
- [33] Bart Goethals. Frequent Set Mining. In *The Data Mining and Knowledge Discovery Handbook*, chapter 17, pages 377–397. Springer-Verlag, 2005. (Cited on page 14.)
- [34] Karam Gouda and Mohammed Javeed Zaki. GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets. *Data Mining and Knowledge Discovery (DMKD)*, 11(3):1–20, 2005. (Cited on page 16.)
- [35] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. DB-CSC: A Density-Based Approach for Subspace Clustering in Graphs with Feature Vectors. In *Proc. of European Conf. on Machine Learning and Princ. and Pract. of Knowledge Discovery in Databases (ECML/PKDD)*, pages 565–580. Springer-Verlag, 2011. (Cited on page 31.)
- [36] Jiawei Han and Micheline Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2 edition, 2005. (Cited on page 3.)
- [37] Jiawei Han, Jian Pei, and Yiwen Yin. Mining Frequent Patterns without Candidate Generation. In *Proc. of Int. Conf. on Management of Data (SIGMOD)*, pages 1–12. ACM, 2000. (Cited on page 15.)

- [38] Daniel Hanisch, Alexander Zien, Ralf Zimmer, and Thomas Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(Suppl 1):145–154, 2002. (Cited on page 30.)
- [39] Leland H Hartwell, John J Hopfield, Stanislas Leibler, and Andrew W Murray. From molecular to modular cell biology. *Nature*, 402(6761 Suppl):C47–52, 1999. (Cited on pages 30 and 83.)
- [40] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for Association Rule Mining - A General Survey and Comparison. *SIGKDD Explor. Newsl.*, 2(1):58–64, 2000. (Cited on page 14.)
- [41] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996. (Cited on page 3.)
- [42] Lars J Jensen, Michael Kuhn, Manuel Stark, Samuel Chaffron, Chris Creevey, Jean Muller, Tobias Doerks, Philippe Julien, Alexander Roth, Milan Simonovic, Peer Bork, and Christian von Mering. STRING 8 - a global view on proteins and their functional interactions in 630 organisms. *Nucleic acids research*, 37(Suppl 1):412–416, 2009. (Cited on page 85.)
- [43] Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L1 Norm*, pages 405–416, 1987. (Cited on pages 28 and 29.)
- [44] Leonard Kaufman and Peter Rousseeuw. *Finding Groups in Data: an introduction to cluster analysis*. Wiley Interscience, 1990. (Cited on page 30.)
- [45] Arijit Khan, Xifeng Yan, and Kun-Lung Wu. Towards proximity pattern mining in large graphs. In *Proc. of Int. Conf. on Management of Data (SIGMOD)*, pages 867–878. ACM Press, 2010. (Cited on page 31.)
- [46] Arno J Knobbe and Eric K Y Ho. Pattern Teams. In *Proc. of European Conf. on Machine Learning and Princ. and Pract. of Knowledge Discovery in Databases (ECML/PKDD)*, pages 577–584. Springer-Verlag, 2006. (Cited on page 18.)
- [47] Jussi M Kumpula, Mikko Kivela, Kimmo Kaski, and Jari Saramaki. A sequential algorithm for fast clique percolation. *Physical Review E*, 78(2):1–8, 2008. (Cited on pages 23 and 25.)
- [48] Johan Leyritz, Stéphane Schicklin, Sylvain Blachon, Céline Keime, Céline Robardet, Jean-François Boulicaut, Jérémy Besson, Ruggero G Pensa, and Olivier Gandrillon. SQUAT: A web tool to mine human, murine and avian SAGE data. *BMC Bioinformatics*, 9(1):1–12, 2008. (Cited on page 85.)
- [49] Zhi Liang, Meng Xu, Maikun Teng, and Liwen Niu. Comparison of protein interaction networks reveals species conservation and divergence. *BMC Bioinformatics*, 7(1):457, 2006. (Cited on page 83.)

- [50] Christoph Lippert, Nino Shervashidze, and Oliver Stegle. Relational models for generating labeled real-world graphs. In *Proc. of Int. Workshop on Mining and Learning with Graphs*, pages 1–3, 2009. (Cited on page 60.)
- [51] Guimei Liu and Limsoon Wong. Effective Pruning Techniques for Mining Quasi-cliques. In *Proc. of European Conf. on Machine Learning and Princ. and Pract. of Knowledge Discovery in Databases (ECML/PKDD)*, pages 33–49. Springer-Verlag, 2008. (Cited on pages 5 and 23.)
- [52] R. Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949. (Cited on page 5.)
- [53] Kazuhisa Makino and Takeaki Uno. New Algorithms for Enumerating All Maximal Cliques. In *Proc. of Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 260–272. Springer-Verlag, 2004. (Cited on page 21.)
- [54] Wagner Jr Meira. *Structural Correlation Pattern Mining for Large Graphs*. PhD thesis, Universidade Federal de Minas Gerais, 2010. (Cited on page 32.)
- [55] Tom M Mitchell. *Machine Learning*. McGraw-Hill, 1997. (Cited on page 28.)
- [56] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23–28, 1965. (Cited on pages 21 and 48.)
- [57] Flavia Moser, Rong Ge, and Martin Ester. Joint cluster analysis of attribute and relationship data without-a-priori specification of the number of clusters. In *Proc. of Int. Conf. on Knowledge discovery and Data Mining (KDD)*, pages 510–519. ACM Press, 2007. (Cited on pages 27 and 29.)
- [58] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining Cohesive Patterns from Graphs with Feature Vectors. In *Proc. of SIAM Int. Conf. on Data Mining (SDM)*, pages 593–604. SIAM, 2009. (Cited on pages 5, 27, 30, 31, 32, 38, and 60.)
- [59] Pierre-Nicolas Mougél, Marc Plantevit, Christophe Rigotti, Olivier Gandrillon, and Jean-François Boulicaut. Constraint-Based Mining of Sets of Cliques Sharing Vertex Properties. In *Proc. of Workshop on Analysis of Complex NETWORKS (ACNE) co-located with ECML/PKDD*, pages 1–14, 2010. (Cited on page 6.)
- [60] Pierre-Nicolas Mougél, Marc Plantevit, Christophe Rigotti, Olivier Gandrillon, and Jean-François Boulicaut. A Data Mining Approach to Highlight Relations Between Functional Modules. In *Proc. of Integrative Post-Genomics (IPG)*, page 1, 2010. (Cited on page 6.)
- [61] Pierre-Nicolas Mougél, Marc Plantevit, Christophe Rigotti, Olivier Gandrillon, and Jean-François Boulicaut. Extraction sous Contraintes d’Ensembles de Cliques Homogènes. In *Proc. of Extraction et Gestion de la Connaissance (EGC)*, pages 443–454, 2011. (Cited on page 6.)

- [62] Pierre-Nicolas Mougel, Christophe Rigotti, and Olivier Gandrillon. Finding Collections of k -Clique Percolated Components in Attributed Graphs. In *Proc. of Pacific-Asia Conf. on Knowl. Discov. and Data Mining (PAKDD)*, pages 181–192. Springer-Verlag, 2012. (Cited on page 6.)
- [63] Pierre-Nicolas Mougel, Christophe Rigotti, and Olivier Gandrillon. Finding Collections of Protein Modules in Protein-Protein Interaction Networks. In *Proc. of Bioinformatics and Computational Biology (BiCOB)*, pages 1–7, 2012. (Cited on page 6.)
- [64] Katarzyna Musial and Krzysztof Juszczyszyn. Properties of Bridge Nodes in Social Networks. In *Proc. of Int. Conf. on Computational Collective Intelligence (ICCCI)*, pages 357–364. Springer-Verlag, 2009. (Cited on page 70.)
- [65] John C Newman and Alan M Weiner. L2L: a simple tool for discovering the hidden significance in microarray expression data. *Genome Biology*, 6(9):1–18, 2005. (Cited on page 87.)
- [66] Mark E J Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167, 2003. (Cited on page 20.)
- [67] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005. (Cited on pages 5, 23, 24, 65, 69, and 70.)
- [68] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society - Supplementary information. *Nature*, 435(7043), 2005. (Cited on page 70.)
- [69] Jayesh Pandey, Mehmet Koyuturk, and Ananth Grama. Functional characterization and topological modularity of molecular interaction networks. *BMC Bioinformatics*, 11(Suppl 1):S35, 2010. (Cited on page 83.)
- [70] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In *Proc. of Int. Conf. on Database Theory (ICDT)*, pages 398–416. Springer-Verlag, 1999. (Cited on pages 5 and 16.)
- [71] Jian Pei, Jiawei Han, and Runying Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In *Proc. of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30. ACM, 2000. (Cited on page 16.)
- [72] Solen Quiniou, Peggy Cellier, Thierry Charnois, and Dominique Legallois. Fouille de graphes sous contraintes linguistiques pour l’exploration de grands textes. In *Proc. of Traitement Automatique des Langues Naturelles (TALN)*, pages 253–266, 2012. (Cited on page 103.)
- [73] Luc De Raedt and Albrecht Zimmermann. Constraint-Based Pattern Set Mining. In *Proc. of SIAM Int. Conf. on Data Mining (SDM)*, pages 237–248. SIAM, 2007. (Cited on pages 18 and 31.)

- [74] Céline Robardet. Constraint-based Pattern Mining in Dynamic Graphs. In *Proc. of IEEE Int. Conf. on Data Mining (ICDM)*, pages 950–955. IEEE Computer Society, 2009. (Cited on page 4.)
- [75] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966. (Cited on page 20.)
- [76] Jouni K Seppänen and Heikki Mannila. Dense itemsets. In *Proc. of Int. Conf. on Knowledge discovery and Data Mining (KDD)*, volume 4, pages 683–688. ACM, 2004. (Cited on page 17.)
- [77] Jun Sese, Mio Seki, and Mutsumi Fukuzaki. Mining networks with shared items. In *Proc. of Int. Conf. on Information and Knowledge Management (CIKM)*, pages 1681–1684. ACM, 2010. (Cited on pages 27 and 31.)
- [78] Arno Siebes, Jilles Vreeken, and Matthijs van Leeuwen. Item Sets That Compress. In *Proc. of SIAM Int. Conf. on Data Mining (SDM)*, pages 393–404. SIAM, 2006. (Cited on page 18.)
- [79] Arlei Silva, Wagner Jr Meira, and Mohammed Javeed Zaki. Structural correlation pattern mining for large graphs. In *Proc. of Workshop on Mining and Learning with Graphs (MLG)*, pages 119–126. ACM, 2010. (Cited on pages 32 and 38.)
- [80] Arlei Silva, Wagner Jr Meira, and Mohammed Javeed Zaki. Mining attribute-structure correlated patterns in large attributed graphs. *Proc. of the VLDB Endowment*, 5(5):466–477, 2012. (Cited on pages 23, 27, and 32.)
- [81] Arnaud Soulet and Bruno Crémilleux. An Efficient Framework for Mining Flexible Constraints. In *Proc. of Pacific-Asia Conf. on Knowl. Discov. and Data Mining (PAKDD)*, pages 661–671. Springer-Verlag, 2005. (Cited on page 18.)
- [82] Xing Sun and Andrew B Nobel. Significance and Recovery of Block Structures in Binary Matrices with Noise. In *Proc. of Conf. on Learning Theory (COLT)*, pages 109–122. Springer-Verlag, 2006. (Cited on page 17.)
- [83] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., 1 edition, 2005. (Cited on page 3.)
- [84] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science (TCS)*, 363(1):28–42, 2006. (Cited on pages 21, 53, and 69.)
- [85] Hanghang Tong, Brian Gallagher, Christos Faloutsos, and Tina Eliassi-rad. Fast Best-Effort Pattern Matching in Large Attributed Graphs. In *Proc. of Int. Conf. on Knowledge discovery and Data Mining (KDD)*, pages 737–746. ACM, 2007. (Cited on page 30.)
- [86] Igor Ulitsky and Ron Shamir. Identification of functional modules using network topology and high-throughput data. *BMC Systems Biology*, 1(1):1–17, 2007. (Cited on pages 30 and 83.)

- [87] Victor E Velculescu, Lin Zhang, Bert Vogelstein, and Kenneth W Kinzler. Serial Analysis of Gene Expression. *Science*, 270(5235): 484–487, 1995. (Cited on page 85.)
- [88] Lei Xu and Irwin King. A PCA approach for fast retrieval of structural patterns in attributed graphs. *IEEE Trans. on Systems, Man, and Cybernetics*, 31(5):812–817, 2001. (Cited on page 30.)
- [89] Ramon Xulvi-Brunet and Hongzhe Li. Co-expression networks: graph properties and topological comparisons. *Bioinformatics*, 26(2):205–214, 2010. (Cited on page 83.)
- [90] Guizhen Yang. The Complexity of Mining Maximal Frequent Itemsets and Maximal Frequent Patterns. In *Proc. of Int. Conf. on Knowledge discovery and Data Mining (KDD)*, pages 344–353. ACM, 2004. (Cited on page 16.)
- [91] Mohammed Javeed Zaki. Scalable Algorithms for Association Mining. *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 12(3):372–390, 2000. (Cited on page 15.)
- [92] Mohammed Javeed Zaki and Mitsunori Ogihara. Theoretical Foundations of Association Rules. In *Proc. of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 1–8. ACM, 1998. (Cited on page 16.)
- [93] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *Proc. of Int. Conf. on Knowledge discovery and Data Mining (KDD)*, pages 797–802. ACM, 2006. (Cited on page 23.)
- [94] Shihua Zhang, Xuemei Ning, and Xiang-sun Zhang. Identification of functional modules in a PPI network by clique percolation clustering. *Computational Biology and Chemistry*, 30(6):445–451, 2006. (Cited on pages 30 and 83.)
- [95] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph Clustering Based on Structural/Attribute Similarities. In *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pages 718–729. Springer-Verlag, 2009. (Cited on page 29.)
- [96] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Clustering Large Attributed Graphs: An Efficient Incremental Approach. In *Proc. of IEEE Int. Conf. on Data Mining (ICDM)*, pages 689–698. IEEE Computer Society, 2010. (Cited on page 29.)