

**MINERAÇÃO DE PADRÕES DE CORRELAÇÃO
ESTRUTURAL EM GRANDES GRAFOS**

ARLEI LOPES DA SILVA

**MINERAÇÃO DE PADRÕES DE CORRELAÇÃO
ESTRUTURAL EM GRANDES GRAFOS**

Dissertação apresentada ao Programa de Pós-Graduação em Computer Science do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Computer Science.

ORIENTADOR: WAGNER MEIRA JR.

Belo Horizonte

Maio de 2011

ARLEI LOPES DA SILVA

**STRUCTURAL CORRELATION PATTERN
MINING FOR LARGE GRAPHS**

Dissertation presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

ADVISOR: WAGNER MEIRA JR.

Belo Horizonte

May 2011

© 2011, Arlei Lopes da Silva.
Todos os direitos reservados.

Silva, Arlei Lopes da
S586m Mineração de Padrões de Correlação Estrutural em
Grandes Grafos / Arlei Lopes da Silva. — Belo
Horizonte, 2011
xxvi, 128 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de
Minas Gerais

Orientador: Wagner Meira Jr.

1. Computação – Teses. 2. Mineração de dados –
Teses. 3. Teoria dos grafos. Teses. I. Orientador.
II. Título.

CDU 519.6*73(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Mineração de padrões de correlação estrutural em grandes grafos

ARLEI LOPES DA SILVA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:



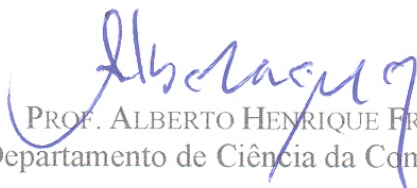
PROF. WAGNER MEIRA JUNIOR
Departamento de Ciência da Computação - UFMG



DR. LOIC PASCAL GILLES CERF
Bolsista de Pós-Doutorado – DCC - UFMG



PROF. MOHAMMED JAVEED ZAKI
Rensselaer Polytechnic Institute



PROF. ALBERTO HENRIQUE FRAIDE LAENDER
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 05 de maio de 2011.

Acknowledgements

First of all, I would like to thank my advisor Wagner Meira Jr. for his collaboration over the past 5 years. I was an undergrad student with average grades when I knocked Meira's door to ask him for a position as an undergrad researcher. Since then, he has provided me with many opportunities to learn the skills I needed to finish this thesis and to progress in my career as a computer scientist.

I would also like to thank Mohammed J. Zaki for his guidance during the 6 months I have spent as a visiting scholar at RPI and for the collaboration that followed. Most of the technical contributions of this thesis were a consequence of my attempts to address some of his comments. Thanks also to Loïc Cerf and Alberto Laender who were part of my thesis committee and contributed to the improvement of this work.

I am honored to have worked with Adriano Pereira, Fabio Figueiredo, Fernando Mourão, Herico Valiati, Jussara Almeida, Leonardo Rocha, Livia Simões, Loïc Cerf, Marco Ribeiro, Marcos Gonçalves, Mariângela Cherchiglia, Mehdi Kaytoue, Nathan Mariano, Odilon Queiroz, Pedro Calais, Walter Santos and Sara Guimarães. I would like to give my special thanks to Adriano for his generous support since my first years as an undergrad researcher.

Being a mastering student is much better when you have some good friends, such as Carlos Teixeira and Charles Gonçalves, around. Thanks also to all my colleagues from the e-Speed Lab, where I have spent a significant part of the past 5 years. In particular, I would like to thank Bruno Coutinho for his always kind helping. Thanks also to the staff from the Computer Science Department, specially Cida and Sônia, for the patience and sympathy, despite of my constant lack of organization.

While I was a high school student at Cidade dos Meninos, many teachers motivated me to study hard to get into college, such as Julio, Adalberto, Katia, and Glauceones. I am also grateful to Jairo Azevedo, founder of Cidade dos Meninos, and the numerous contributors that support the APHDP.

During most of my undergraduate studies, I was assisted by the community from Universidade Federal de Minas Gerais through Fundação Mendes Pimentel.

I have always received support and encouragement from my family and friends, specially my mother Ivone, my brother Andre, my friend David and my brother Alex, without whom I would never get anywhere close to this.

“Dubium sapientiae initium.”
(Latin proverb)

Resumo

Grafos têm se estabelecido como um poderoso arcabouço teórico para a modelagem de interações em cenários variados. Enquanto a disponibilidade de dados em larga escala motivou o desenvolvimento de tal arcabouço, o enriquecimento desses dados guia a pesquisa em grafos na direção de novos métodos capazes de explorar essa riqueza de forma útil. Uma representação estendida interessante de grafos é a de grafos com atributos nos vértices. Atributos de vértices desempenham um papel importante em diversos grafos reais. Além disso, sabe-se que, em muitos desses grafos, vértices se organizam naturalmente como subgrafos densos. Tais subgrafos possuem significado relevante em diversos grafos reais, sendo denominados comunidades em redes sociais e identificando complexos proteicos em redes de proteínas, dentre outras aplicações.

Neste trabalho, estudamos a correlação entre conjuntos de atributos e a formação de subgrafos densos, o que denominamos mineração de padrões de correlação estrutural. A correlação estrutural mede como um conjunto de atributos induz subgrafos densos em grafos com atributos. Um padrão de correlação estrutural é um subgrafo denso induzido por um conjunto de atributos em particular. Modelamos padrões de correlação estrutural em termos de padrões de mineração de dados existentes. Com base em tal modelagem, propomos técnicas de normalização que avaliam o quanto a correlação estrutural de um conjunto de atributos desvia do esperado. Além disso, propomos algoritmos eficientes e escaláveis para a mineração de padrões de correlação estrutural.

Nós mostramos que a mineração de padrões de correlação estrutural é capaz de prover conhecimento relevante sobre a relação entre conjuntos de atributos e subgrafos densos em grafos reais. Em particular, aplicamos os algoritmos propostos na correlação entre palavras-chave associadas a pesquisadores e a formação de grupos de pesquisa em redes de colaboração, no estudo de comunidades induzidas pelo gosto musical em uma rede social, na análise de como grupos conectados de artigos emergem em torno de tópicos de pesquisa em uma rede de citação, e na avaliação da relação entre expressão e funcionalidade em uma rede de interação proteica. Também avaliamos o desempenho de tais algoritmos, verificando que eles possibilitam a análise de grandes bases de dados.

Abstract

Graphs have been established as a powerful theoretical framework for modeling several types of interactions in a variety of scenarios. While the availability of large scale data led to the development of a framework for large scale graph analysis, the enrichment of such data drives the graph research to new methods able to explore such richness in a useful manner. An interesting extended graph representation is called attributed graph. Vertex attributes play an important role in several real life graphs. Moreover, it is broadly known that in several of these graphs vertices are organized into dense subgraphs. Such subgraphs have a relevant meaning in several real life graphs, being called communities in social networks and identifying protein complexes in protein-protein interaction networks.

In this work, we study the correlation between attribute sets and the formation of dense subgraphs in large attributed graphs, which we call structural correlation pattern mining. The structural correlation measures how a set of attributes induces dense subgraphs in attributed graphs. A structural correlation pattern is a dense subgraph induced by a particular attribute set. We model the structural correlation pattern mining in terms of existing data mining patterns. Based on such definitions, we propose normalization approaches in order to assess how the structural correlation of a given attribute set deviates from the expected. Moreover, we propose efficient and scalable algorithms for structural correlation pattern mining.

We show that the structural correlation pattern mining is able to provide relevant knowledge about the relation between attribute sets and dense subgraphs in real attributed graphs. In particular, we apply the proposed algorithms to the correlation between keywords associated with researchers and the formation of research groups in collaboration networks, in the study of communities induced by musical taste in a social network, in the analysis of how well connected groups of papers emerge around research topics in a citation network, and in the evaluation of the relation between expression and functionality in a PPI network. We also evaluate the performance of such algorithms, verifying that they enable the analysis of large datasets.

List of Figures

1.1	Illustrative example graph	2
1.2	Co-authorship graph extracted from the DBLP digital library	2
1.3	Dense subgraph from the graph shown in Figure 1.1	5
1.4	Dense subgraph from the graph shown in Figure 1.1	6
1.5	Dense subgraph from the real graph shown in Figure 1.2	7
2.1	Set Enumeration Tree	16
2.2	Subgraph of the 0.6-quasi-clique shown in Figure 1.3 which is not a 0.6-quasi-clique	16
2.3	Complete lattice for the set of attributes $\{A, B, C, D, E\}$	24
3.1	Graph from Figure 1.1, vertices 3 to 11 have the attribute A and are in dense subgraphs, vertices 1 and 2 have the attribute A but are out of dense subgraphs	29
3.2	Graph from Figure 1.1, vertices 1, 3, and 6 have the attribute C and the other vertices do not have C	30
3.3	Graph from Figure 1.1, vertices 6 to 11 have the attribute set $\{A, B\}$ and are in dense subgraphs, the other vertices do not have $\{A, B\}$	30
3.4	Graph induced by the attribute set $\{search, rank\}$ from the collaboration graph shown in Figure 1.2.	31
3.5	Clique with the same number of vertices that the graph shown in Figure 1.1	33
4.1	Order of visit of candidate patterns in a BFS search	48
4.2	Order of visit of candidate patterns in a DFS search	48
5.1	Cumulative degree, attribute frequency, and attributes per vertex distribution (in log-log scale) for the DBLP dataset	75
5.2	Expected structural correlation computed using the simulation model (sim- ϵ_{exp}) and the analytical model $\max\text{-}\epsilon_{exp}$	77

5.3	Graph induced by the attribute set $\{search, rank\}$ in the DBLP dataset	78
5.4	Structural correlation pattern induced by the attribute set $\{search, rank\}$, size = 13, and $\gamma=0.58$	79
5.5	Structural correlation pattern induced by the attribute set $\{perform, system\}$, size = 37, and $\gamma=0.5$	80
5.6	Inverse cumulative structural coverage (κ), structural correlation (ϵ), and normalized structural correlation (δ_2)	80
5.7	Inverse cumulative degree, attribute frequency, and attributes per vertex distribution (in log-log scale) for the LastFm dataset	82
5.8	Expected structural correlation computed using the simulation model (sim- ϵ_{exp}) and the analytical model max- ϵ_{exp}	84
5.9	Graph induced by the attribute set $\{Sufjan Stevens, Wilco\}$ in the LastFm dataset	85
5.10	Structural correlation pattern induced by the attribute set $\{Sufjan Stevens, Wilco\}$, size = 11, and $\gamma=0.7$	86
5.11	Structural correlation pattern induced by the attribute set $\{Van Morrison\}$, size = 34, and $\gamma=0.52$	86
5.12	Inverse cumulative structural coverage (κ), structural correlation (ϵ), and normalized structural correlation (δ_2)	87
5.13	Inverse cumulative degree, attribute frequency, and attributes per vertex distributions (in log-log scale) for the CiteSeer dataset	89
5.14	Expected structural correlation computed using the simulation model (sim- ϵ_{exp}) and the analytical model max- ϵ_{exp}	90
5.15	Graph induced by the attribute set $\{node, wireless\}$ in the CiteSeer dataset	91
5.16	Structural correlation pattern induced by the attribute set $\{node, wireless\}$, size = 9, and $\gamma=0.5$ (EEBAWN - Energy-efficient Broadcasting in All-wireless networks, SAEECATMAWN - Span: An Energy-efficient Coordination Algorithm for Topology Maintenance in Ad-hoc Wireless Networks, ECRWAN - Energy-conserving Routing in Wireless Ad-hoc Networks, MEMWN - Minimum-Energy Mobile Wireless Networks, CMCP-WSN - Connectivity Maintenance and Coverage Preservation in Wireless Sensor Networks, GIECAR - Geography-informed Energy Conservation for Ad-hoc Routing, MEBAWN - Minimum-energy Broadcast in All-wireless Networks, DTCWSNDCPEOMWAN - Distributed Topology Control in Wireless Sensor Networks with Distributed Control for Power-efficient Operation in Multihop Wireless Ad-hoc Networks)	92

5.17	Structural correlation pattern induced by the attribute set $\{perform, system\}$, size = 21, and $\gamma=0.5$ (AC - Attribute Caches, SLCM - Systems for Late Code Modification, OTDRN - Observing TCP Dynamics in Real Networks, TCIG - Transparent Controls for Interactive Graphics, LILP - Limits of Instruction Level Parallelism, DLMR - DECWRL/Livermore Magic Release, LTOAC6BA - Link-time Optimization of Address Calculation on a 64-bit Architecture, LTCM - Link-time Code Modification, MSDCDSP - Memory-system Design Considerations for Dinamically Scheduled Processors, BMFCEG - Boolean Matching for Full-custom ECL Gates, CWPP - Cache Write Policies and Performance, SMCM - Shared Memory Consistency Models, CEFCGASDSM - Comparative Evaluation of Fine and Coarse-grain Approaches for Software Distributed Shared Memory, RLG - Recursive Layout Generation, EDPP - Efficient Dynamic Procedure Placement, DSDI - Drip: A Schematic Drawing Interpreter, FPPDMA - Fluoroelastomer Pressure Pad Design for Microelectronic Applications, TTOCC - Trade-offs in Two-level on-chip Caching, IOCCIOCD - I/O Component Characterization for I/O Cache Designs, VMFS - Virtual Memory vs File System, EWWWWC - Experience with a Wireless World Wide Web Client)	93
5.18	Inverse cumulative structural coverage (κ), structural correlation (ϵ), and normalized structural correlation (δ_2)	94
5.19	Inverse cumulative degree, attribute frequency, and attributes per vertex distributions (in log-log scale) for the Human dataset	96
5.20	Graph induced by the attribute set $\{+SHCN086, +SHCN087, +SHCN088\}$ in the Human dataset	97
5.21	Structural correlation pattern induced by the attribute set $\{+SHCN086, +SHCN087, +SHCN088\}$, size = 7, and $\gamma=0.5$	98
5.22	Expected structural correlation computed using the simulation model (sim- ϵ_{exp}) and the analytical model $\max\text{-}\epsilon_{exp}$	99
5.23	Cumulative structural coverage (κ), structural correlation (ϵ), and normalized structural correlation (δ_2)	100
5.24	Parameter sensitivity w.r.t the minimum structural correlation (ϵ_{min})	102
5.25	Parameter sensitivity w.r.t the minimum normalized structural correlation (δ_{min})	103
5.26	Runtime of SCPM-BFS , Naive and SCPM-DFS w.r.t. γ_{min}	105
5.27	Runtime of SCPM-BFS , Naive and SCPM-DFS w.r.t. min_size	105
5.28	Runtime of SCPM-BFS , Naive and SCPM-DFS w.r.t. σ_{min}	106

5.29	Runtime of SCPM-BFS , Naive and SCPM-DFS w.r.t. ϵ_{min}	106
5.30	Runtime of SCPM-BFS , Naive and SCPM-DFS w.r.t. δ_{min}	107
5.31	Runtime of SCPM-DFS , SCPM-BFS-SAMP and SCPM-DFS-SAMP w.r.t. θ_{max}	109
5.32	Mean squared error of SCPM-SAMP-DFS w.r.t. θ_{max}	109
5.33	Runtime of the SCPM-DFS and the Naive algorithm w.r.t. k	110
5.34	Runtime and speedup of PAR-SCPM-DFS w.r.t. the number of cores .	111
5.35	Runtime and speedup of PAR-SCPM-BFS w.r.t. the number of cores .	112
5.36	Runtime and speedup of the PAR-SAMP-SCPM-BFS w.r.t. the number of cores	112
5.37	Runtime and speedup of the PAR-SAMP-SCPM-DFS w.r.t. the number of cores	113

List of Tables

1.1	Attributes of the vertices from the graph shown in Figure 1.1	5
2.1	Frequent attribute sets from the vertex attributes shown in Table 1.1 (the minimum support set is 3)	24
3.1	Table of Symbols	29
3.2	Number of possible settings, expected structural correlation (ϵ_{exp}), simulation-based structural correlation ($\text{sim-}\epsilon_{exp}$), and analytical normalized structural correlation ($\text{max-}\epsilon_{exp}$) for N randomly selected vertices from the graph 1.1	34
3.3	Attribute sets, with their respective values of σ , κ , and ϵ , from the graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1 if $\sigma_{min} = 3$, $\gamma_{min} = 0.6$, $min_size = 4$, and $\epsilon_{min} = 0.5$	40
3.4	Structural correlation patterns, with their respective sizes (size) and densities γ , from the graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1 if $\sigma_{min} = 3$, $\gamma_{min} = 0.6$, $min_size = 4$, and $\epsilon_{min} = 0.5$	41
3.5	Attribute sets, with their respective values of σ , κ , ϵ , $\text{max-}\epsilon_{exp}$, and δ_2 , from the graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1 if $\sigma_{min} = 3$, $\gamma_{min} = 0.6$, $min_size = 4$, and $\delta_{min} = 1.0$	42
3.6	Structural correlation patterns, with their respective sizes (size) and densities γ , from the graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1 if $\sigma_{min} = 3$, $\gamma_{min} = 0.6$, $min_size = 4$, and $\delta_{min} = 1$	42
5.1	Top- ϵ attribute sets from DBLP	76
5.2	Top- δ_2 attribute sets from DBLP	77

5.3	Scatter plots of the correlations between support (σ), structural coverage (κ), structural correlation (ϵ), and normalized structural correlation for different attribute set sizes ($ S $) in the DBLP dataset	81
5.4	Top- ϵ attribute sets from the LastFm dataset	83
5.5	Top- δ_2 attribute sets from the LastFm dataset	84
5.6	Scatter plots of the correlations between support (σ), structural coverage (κ), structural correlation (ϵ), and normalized structural correlation for different attribute set sizes ($ S $) in the LastFm dataset	88
5.7	Top- ϵ attribute sets from CiteSeer	90
5.8	Top- δ_2 attribute sets from the CiteSeer dataset	91
5.9	Scatter plots of the correlations among support (σ), structural coverage (κ), structural correlation (ϵ), and normalized structural correlation for different attribute set sizes ($ S $) in the CiteSeer dataset	95
5.10	Top- ϵ attribute sets from the Human dataset	97
5.11	Top- δ_2 attribute sets from the Human dataset	100
5.12	Scatter plots of the correlations among support (σ), structural coverage (κ), structural correlation (ϵ), and normalized structural correlation for different attribute set sizes ($ S $) in the Human dataset	101

List of Algorithms

1	Set Enumeration Tree Algorithm	18
2	extend	19
3	Simulation Null Model for the Structural Correlation Algorithm	35
4	Naive Algorithm For Structural Correlation Pattern Mining	47
5	structural-correlation	49
6	coverage-BFS	49
7	coverage-DFS	50
8	structural-correlation-with-sampling	54
9	check-vertex-in-quasi-clique	55
10	find-quasi-clique-BFS	55
11	find-quasi-clique-DFS	56
12	top-k-structural-correlation-patterns	56
13	try-to-update-top-patterns	58
14	SCPM Algorithm	59
15	enumerate-patterns	60
16	par-structural-correlation	62
17	par-coverage-BFS	63
18	par-coverage-DFS	64
19	par-coverage	65
20	par-check-vertex-in-quasi-clique	66
21	par-find-quasi-clique-BFS	67
22	par-find-quasi-clique-DFS	68
23	par-find-quasi-clique	69
24	par-top-k-structural-correlation-patterns	69
25	par-top-k-scps-thread	70
26	par-top-k-scps-iteration	71
27	par-try-to-update-top-patterns	72

Contents

Resumo	xiii
Abstract	xv
List of Figures	xvii
List of Tables	xxi
1 Introduction	1
1.1 Structural Correlation Pattern Mining	4
1.2 Contributions of This Work	8
1.3 Outline	9
2 Background and Related Work	11
2.1 Dense Subgraphs	11
2.2 Quasi-clique Mining	14
2.2.1 Vertex Pruning	17
2.2.2 Candidate Quasi-clique Pruning	18
2.3 Frequent Itemset Mining	22
2.4 Related Work	25
3 Structural Correlation Pattern Mining: Definitions	27
3.1 Structural Correlation	28
3.2 Normalized Structural Correlation	32
3.3 Structural Correlation Patterns	37
3.4 Structural Correlation Pattern Mining Problem	38
4 Structural Correlation Pattern Mining: Algorithms	45
4.1 Naive Algorithm	46
4.2 Computing the Structural Correlation	47

4.3	Pruning Techniques	51
4.4	Sampling	53
4.5	Top-k Structural Correlation Patterns	57
4.6	The SCPM Algorithm	60
4.7	Parallel Algorithms	61
4.7.1	Computing the Structural Correlation	62
4.7.2	Sampling	65
4.7.3	Top-k Structural Correlation Patterns	68
5	Experimental Evaluation	73
5.1	Case Studies	74
5.1.1	DBLP	75
5.1.2	LastFm	82
5.1.3	CiteSeer	88
5.1.4	Human	96
5.2	Parameter Sensitivity and Setting	102
5.3	Performance Evaluation	103
5.3.1	Computing the Structural Correlation	104
5.3.2	Sampling	107
5.3.3	Discovering the Top-K Structural Correlation Patterns	109
5.3.4	Parallel Algorithms	110
5.3.5	Discussion	113
6	Conclusions	115
6.1	Summary of Contributions	115
6.2	Limitations	116
6.3	Future Work	117
	Bibliography	121

Chapter 1

Introduction

Graphs, or *networks*, have been established as a powerful theoretical framework for modeling several types of interaction in a variety of scenarios. Due to its broad applicability, graphs have attracted a great interest from a wide research community, including mathematicians, physicists, sociologists, biologists, and computer scientists. The availability of large real graphs in the last years motivated a broad spectrum of research on the properties of such graphs. Moreover, the combination of new algorithms and powerful hardware have enabled the discovery of complex and interesting patterns from large graphs.

A graph is usually defined as a set of *vertices* (or nodes) connected by a set of *edges*. Figure 1.1 shows an illustrative example of a graph with 11 vertices (1-11) and 21 edges. Figure 1.2 is a real graph extracted from a digital library. Each vertex represents an author and two vertices are connected if their respective authors have already collaborated on a paper. From social networks to food webs, from distribution networks to the WWW, many relevant systems can be modeled through graphs.

The study of graphs has evolved significantly since the solution of the Konigsberg problem by Euler in 1735, which is considered the first scientific work on graphs in the literature. While the first studies were focused on graph theory, subsequent studies applied graphs to the analysis of small social networks. The seminal Milgram's paper [Travers and Milgram, 1969], for example, showed that randomly selected individuals from Boston and Nebraska were connected to randomly selected people from Massachusetts through a connected chain of acquaintances of average size 5.2, what is considered the first evidence of the so called small world phenomenon. However, along the past few years, the study of graphs has witnessed a new shift in the direction of the analysis of large scale statistical properties of real graphs [Newman, 2003]. An extensive study of several real graphs from different domains have shown that interesting

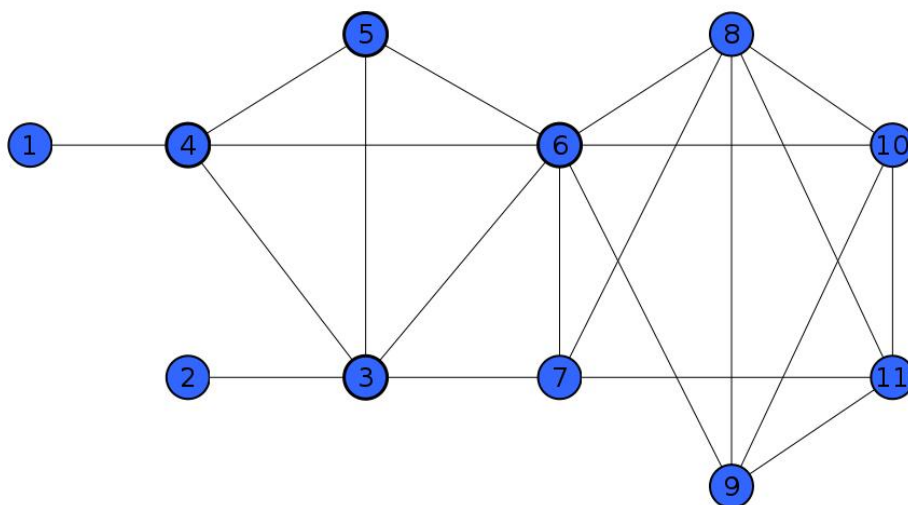


Figure 1.1: Illustrative example graph

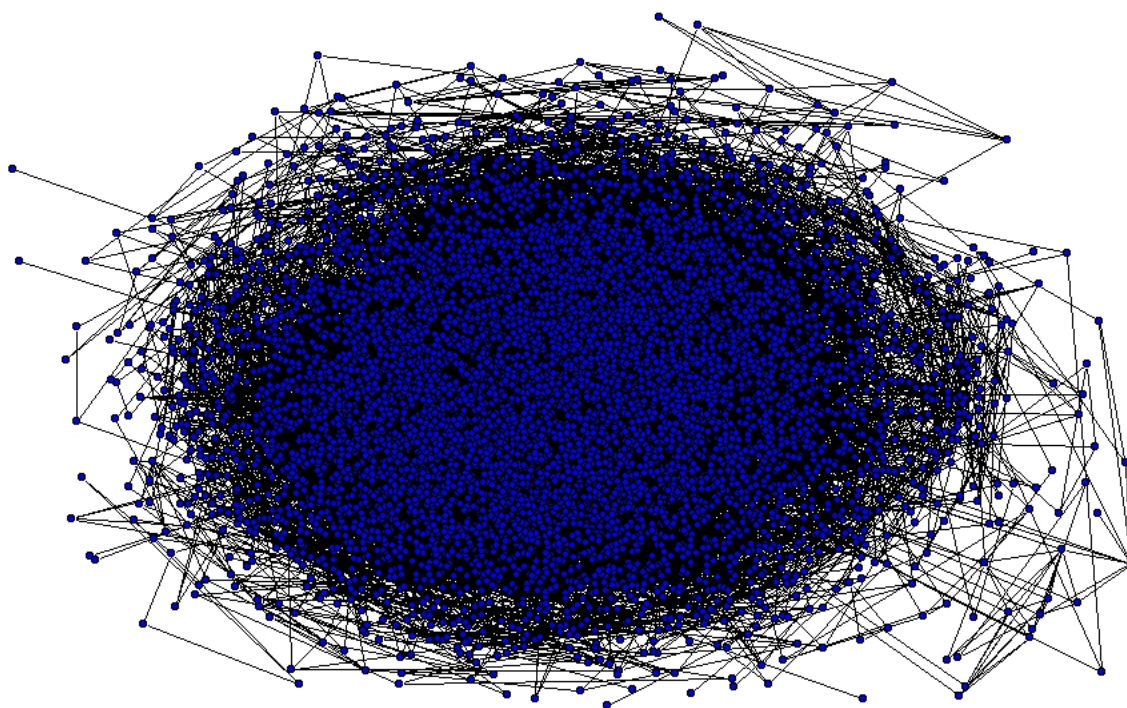


Figure 1.2: Co-authorship graph extracted from the DBLP digital library

properties (e.g., power-law degree distributions, high clustering coefficient, community structure, and small diameter) are common to many of these graphs.

But what is the future of the graph research? Which graph problems and applications will challenge the research community in the upcoming time? Answering such thought-provoking questions is essential for developing long term innovative research on

graphs. A significant part of the research community on graphs has agreed that the graph research is evolving towards more complex models and analysis [Newman, 2003; Leskovec, 2008; Chakrabarti and Faloutsos, 2006].

Along its two centuries of history, the study of graphs has shifted from theory to application, and then to large scale. However, a simple undirected graph (see Figure 1.1) has remained as the standard model on graph research. Such a simple model is popular for two basic reasons: (1) it is powerful enough for the analysis of topological static properties of graphs (e.g., degree distribution, clustering coefficient) and (2) it is simple for both understanding and computational processing. Nevertheless, in recent years, the availability of rich data from complex scenarios has motivated analyses that are far beyond static topological properties.

Extending the standard simple graph framework is, in general terms, the topic of this work. While the availability of large scale data led to the development of a framework for large scale graph analysis, the availability of rich data drives the graph research to new methods able to explore such data in a useful manner. Graphs with directed, weighted, multi-typed, and attributed edges, for example, may provide novel interesting knowledge in several application scenarios. Similarly, vertices may have multiple types and attributes. Moreover, some scenarios require the study of multiple graphs or even several snapshots of the same graph, instead of a single static graph. Extracting useful knowledge from such complex large datasets brings new challenges not only in terms of modeling but also efficiency and scalability. In order to address such challenges, a new branch of *data mining* known as *graph mining* has attracted great attention of the data mining community in recent years.

Data mining is a discipline dedicated to the extraction of useful knowledge from large databases [Witten and Frank, 2005; Han, 2005; Tan et al., 2005; Fayyad et al., 1996]. Such knowledge is usually expressed in the form of *patterns*, which are structures for describing and predicting properties of the data [Hand et al., 2001]. Examples of patterns include association rules, clusters, and outliers. Data mining is considered part of the knowledge discovery process, which also includes data cleaning, data selection, data transformation, pattern evaluation, and knowledge presentation. Graph mining is the discovery of knowledge from graph databases. Graph mining tasks, such as frequent subgraph mining, dense subgraph discovery, and motif detection, have great applications in many fields. Recent research efforts on mining large complex graph databases have led to the definition of innovative new patterns able to extract useful knowledge from extended graph representations in an efficient and scalable fashion. The next section introduces the problem studied in this work, which we call structural correlation pattern mining, and may be considered one of such efforts.

1.1 Structural Correlation Pattern Mining

This work studies the problem of correlating vertex attributes and the existence of dense subgraphs in an attributed graph (i.e., a graph where vertices have attributes). We call this problem *structural correlation pattern mining*, since it correlates vertex attributes and a structural (or topological) information from graphs.

Vertex attributes play an important role in several real life graphs. In social networks, vertex attributes are useful to represent personal characteristics (e.g., age, gender, interests). In protein-protein interaction networks, vertex attributes can represent expression or annotation data. Moreover, vertex attributes can be associated to content (e.g., keywords, tags) in the web graph. Table 1.1 shows attributes of vertices from the illustrative example graph shown in Figure 1.1. Each attribute is represented through a letter in the interval A-E. Attributes for vertices shown in Figure 1.2 can be keywords associated to each researcher and can represent their topics of interest.

It is broadly known that in several real graphs vertices are naturally organized into dense subgraphs [Fortunato, 2010]. Dense subgraphs are sets of vertices with high cohesion (i.e., strong connections among themselves). Such subgraphs carry an important meaning in several real life scenarios. In social networks, people interact more intensely inside communities, what is of great interest in social sciences [Newman, 2003; Scott, 2000; Carrington, 2005]. Web pages usually contain links to other related pages, resulting in cyber-communities of pages and sites sharing a common interest [Albert et al., 1999; Dourisboure et al., 2009]. Densely connected proteins in protein-protein interaction networks define molecular complexes that are useful for functional annotation [Spirin and Mirny, 2003]. Groups of publications citing each other can define the related work around a established research topic [Shi et al., 2010]. Detecting and analyzing dense subgraphs has been a long term research problem.

Figures 1.3 and 1.4 are two examples of dense subgraphs identified from the graph shown in Figure 1.1. The subgraph from Figure 1.3 is a *clique* (i.e., a set of vertices where there is an edge between each pair of vertices). A clique is the densest possible subgraph for a given set of vertices. The subgraph shown in Figure 1.4 is not a clique, but it has a high density (or cohesion), since only three pairs of vertices are not adjacent. Figure 1.5 is a dense subgraph extracted from the graph shown in Figure 1.2, it represents a group of researchers with several internal collaborations.

Most of the traditional dense subgraph identification algorithms, as well as many other algorithms for graphs, rely only on topological information (i.e., vertices and edges). However, in many real life scenarios the relationships among entities is only one of the sources of information available. Combining different sources of information in

graph analysis has being argued to enable the discovery of novel interesting knowledge from graphs, specially in the data mining and the bioinformatics literature [Silva et al., 2010, 2012; Ge et al., 2008; Moser et al., 2007, 2009; Pei et al., 2005; Zeng et al., 2006; Zhou et al., 2009].

vertex	attributes
1	A, C
2	A
3	A, C, D
4	A,D
5	A, E
6	A, B, C
7	A, B, E
8	A, B
9	A, B
10	A, B, D
11	A, B

Table 1.1: Attributes of the vertices from the graph shown in Figure 1.1

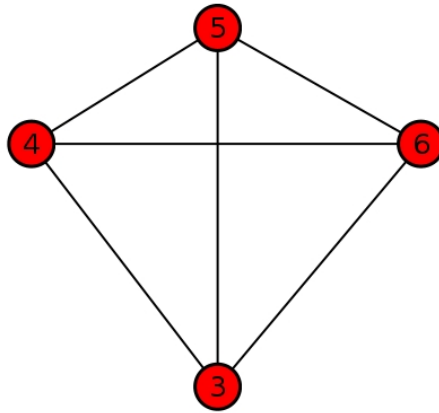


Figure 1.3: Dense subgraph from the graph shown in Figure 1.1

This work studies the correlation between vertex attributes and the formation of dense subgraphs, which we call *structural correlation*. The structural correlation of attribute sets is based on the topology of the graphs induced by them. Attributes with high structural correlation induce sets of vertices that are well connected among themselves in the graph. The structural correlation generalizes the concept of *social correlation*, defined by a previous work [Anagnostopoulos et al., 2008]. While the social correlation is the co-occurrence of an attribute for two adjacent nodes in a social network, the structural correlation considers co-occurrences of single or multiple attributes

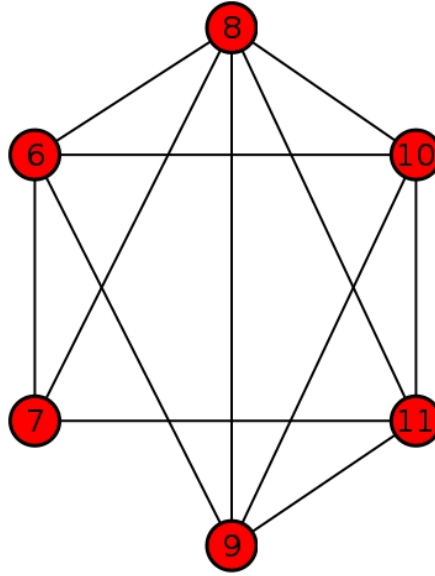


Figure 1.4: Dense subgraph from the graph shown in Figure 1.1

inside dense subgraphs. The graph induced by an attribute set is composed by the vertices that have such attribute set and the edges between these vertices. The structural correlation of a given attribute set is the probability of a vertex to be member of a dense subgraph inside the graph induced by such an attribute set.

Considering the graph shown in Figure 1.1, the vertex attributes shown in Table 1.1 and the dense subgraphs shown in Figures 1.3 and 1.4, the structural correlation of the attribute C is 0, since there is no dense subgraph inside the graph induced by the attribute C (i.e., vertices 1, 3, and 6). The structural correlation of the attribute set $\{A,B\}$ is 1, due to the fact that every vertex is a member of a dense subgraph in the graph induced by $\{A,B\}$. From the collaboration graph shown in Figure 1.2, taking the keywords from the title of the papers of each researcher as their attributes, and considering a dense subgraph a vertex set in which each vertex is connected to, at least, half of its other members, the structural correlation of the keyword “search” is 0.25 (i.e., 25% of the authors with the keyword “search” are members of dense subgraphs composed of other authors who also have this keyword). Therefore, the structural correlation can measure how keywords induce research groups in a collaboration graph.

While the structural correlation is a property of an attribute set, a *structural correlation pattern* is a pair (attribute set, dense subgraph). The pair $(\{A,B\}, \{6,7,8,9,10,11\})$ is an example of a structural correlation pattern from the graph shown in Figure 1.1. Considering the collaboration graph from Figure 1.2, all the authors in the dense subgraph shown in Figure 1.5 have the keywords “search” and “web”. There-

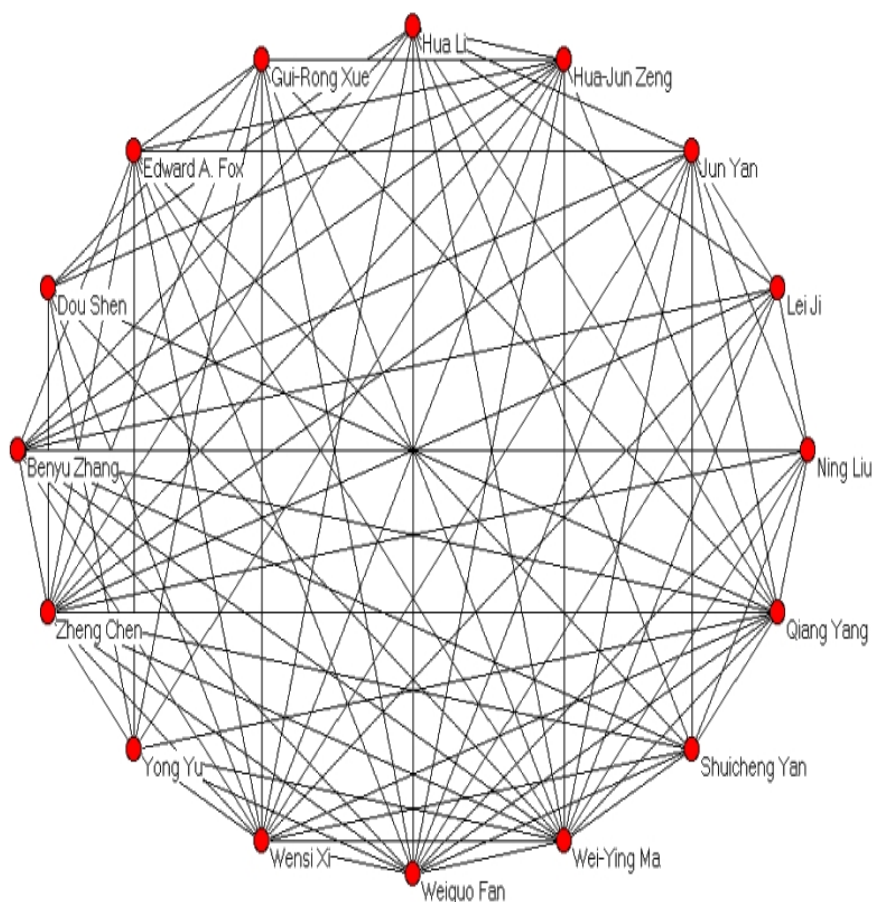


Figure 1.5: Dense subgraph from the real graph shown in Figure 1.2

fore, such a pattern may represent a research group related to the topic *web search*.

Both the structural correlation function and the structural correlation pattern definitions are based on two related hypotheses:

1. Attributes produce dense subgraphs;
2. Dense subgraphs affect the attributes of their internal vertices.

In other words, if none of the two hypotheses are true for a specific attributed graph, the correlation between attribute sets and the existence of dense subgraphs is random. In the specific context of social networks, both these hypotheses are well accepted by the research community. The first hypothesis is related to the *homophily phenomena* that is the tendency of similar people to be connected. The second hypothesis is related to the *social influence*, which is the power of connected people to affect the characteristics and behaviors of each other.

The correlation between attribute sets and the existence of dense subgraphs is not expected to be completely deterministic or completely random. Therefore, it is important to provide interestingness measures for the structural correlation based on how a given value of correlation deviates from the expected. In this work, we propose two null models for the structural correlation function. Such models provide the expected structural correlation for a given attribute set assuming that attributes are set to vertices randomly.

We define the structural correlation pattern mining in terms of two existing data mining patterns: frequent itemsets and quasi-cliques. Frequent itemsets are applied in order to deal with the large number of possible attribute sets in real attributed graphs and quasi-cliques are used as a definition for dense subgraphs. In order to provide efficient and scalable algorithms for the proposed problems we combine existing strategies for frequent itemset and quasi-clique mining with new pruning, sampling and parallelization strategies specific for the structural correlation.

The structural correlation pattern mining constitutes a new graph analysis technique for attributed graphs. It provides knowledge at the level of attribute sets, measuring how they are correlated to the existence of dense subgraphs in the graph, and also at the level of the subgraphs induced by the attribute sets (i.e., the structural correlation patterns). Such a knowledge may be applied to many real graphs from several domains, including social, information, and biological networks.

1.2 Contributions of This Work

The main contributions of this work are summarized as follows:

- **Problem statement:** We introduce the general problem of correlating vertex attributes and the existence of dense subgraphs in attributed graphs. As far as we know, there is no previous work on this problem in the literature.
- **Modeling:** Based on the statement of the problem, we define a structural correlation function and a structural correlation pattern combining two existing data mining patterns (frequent itemsets and quasi-cliques).
- **Algorithm Design:** We design and implement algorithms for structural correlation pattern mining. The algorithms explore pruning, sampling and parallelization as strategies in order to compute the structural correlation and identify structural correlation patterns efficiently.

- **Application and evaluation:** We apply the structural correlation pattern mining to several real datasets from different domains. Using such datasets, we evaluate the performance of the proposed algorithms. We also conduct case studies in order to show the applicability of the structural correlation pattern mining in real-life scenarios.

1.3 Outline

The remaining of this work is organized as follows:

- **Chapter 2 [Background]:** Summarizes existing knowledge related to the structural correlation pattern mining: dense subgraphs, quasi-clique mining, and frequent itemset mining. It also gives an overview on related work in the literature.
- **Chapter 3 [Structural Correlation Pattern Mining: Definitions]:** Gives formal definitions for the structural correlation function and structural correlation pattern mining.
- **Chapter 4 [Structural Correlation Pattern Mining: Algorithms]:** Describes algorithms for structural correlation pattern mining.
- **Chapter 5 [Experimental Evaluation]:** Presents case studies of the application of the structural correlation pattern mining in real-life scenarios and also an experimental evaluation of the proposed algorithms in terms of performance
- **Chapter 6 [Conclusions]:** Discusses the main conclusions and limitations of the proposed work and also points out some future research directions.

Chapter 2

Background and Related Work

In this chapter we present an overview on important topics related to the structural correlation pattern mining: dense subgraphs and the quasi-clique and frequent itemset mining problems. Moreover, we discuss existing work related to the structural correlation pattern mining from the literature. The structural correlation pattern mining correlates attribute sets and the existence of dense subgraphs in an attributed graph. While the concept of attribute set is quite simple (a set of attributes), there is no single formal definition for a dense subgraph in the literature. There are several definitions for dense subgraphs and we survey some of them in Section 2.1. Among the existing definitions for dense subgraphs, we selected the quasi-clique definition. Section 2.2 describes the quasi-clique mining problem and present many existing pruning techniques for the efficient identification of quasi-cliques in graphs. Understanding such techniques is important, since we employ them in the algorithms presented in Chapter 4. In Section 2.3, we overview the frequent itemset mining problem. We apply frequent itemset mining in order to select frequent attribute sets in the structural correlation pattern mining. Section 2.4 discusses related work on the structural correlation pattern mining.

2.1 Dense Subgraphs

The *dense subgraph discovery* is a multidisciplinary problem with great importance in physics, sociology, biology and computer science. Graphs can represent several complex real systems and understanding how vertices are organized into dense subgraphs has been a popular research topic. In social networks, where vertices represent people and edges represent social relationships (e.g., friendship, marriage, co-working), dense subgraphs are of special interest, since they constitute communities [Scott, 2000; Carrington, 2005]. The World Wide Web can also be modeled as a graph, where pages

interact through hyperlinks. Dense subgraphs in the web graph are groups of pages sharing topic similarities (a.k.a. cyber-communities) [Dourisboure et al., 2009]. Moreover, dense subgraphs can also represent an effort in order to enhance the PageRank [Brin and Page, 1998] of web pages artificially [Gyöngyi and Garcia-Molina, 2005]. Similar to web pages, publications can be connected through citations and dense subgraphs are useful for the discovery of topic-related publications. In biology and bioinformatics, the study of protein-protein interaction (PPI) networks is of special interest [Spirin and Mirny, 2003]. Proteins interact in biological processes in the cells and dense subgraphs correspond to functional groups (i.e., proteins with similar functions) in this scenario.

The concepts of dense subgraph and *graph community* (a.k.a graph cluster) are close related. A community is usually defined as a set of vertices significantly more connected among themselves than with vertices outside it [Girvan and Newman, 2002; Newman, 2003; Puig-Centelles et al., 2008; Newman, 2004; Fortunato, 2010]. Therefore, a community is a dense subgraph, but a dense subgraph is not necessarily a community, because dense subgraphs can also have a strong cohesion with the rest of the graph. Such distinction between dense subgraphs and communities is not clear in the literature and the use of both concepts interchangeably is not uncommon. In general, dense subgraphs are exact definitions while communities are based on heuristics. Moreover, most of the algorithms for community identification restrict each vertex to be member of a single community (i.e., such algorithms identify graph partitions).

In this work, we apply a specific dense subgraph identification method (see Section 2.2) in order to compute the correlation between attribute sets and the formation of dense subgraphs. However, it is known that the definition of a dense subgraph may depend on the specific application scenario [Fortunato, 2010]. Although we try to generalize the concept of dense subgraph as much as possible, it is important to notice that some definitions of dense subgraphs can not be applied to our study. We define two requirements for a dense subgraph identification method to be applied to the correlation between attribute sets and the formation of dense subgraphs:

1. **Detecting overlapping dense subgraphs:** It is known that, in real graphs, vertices are shared between dense subgraphs. In social networks, for example, individuals can be members of different social circles. Restricting vertices to be members of a single dense subgraph leads to the neglect of potentially relevant information [Fortunato, 2010].
2. **Allowing vertices not to be assigned to any dense subgraph:** Several algorithms for dense subgraph identification enforce vertices to be set to, at least, a single dense subgraph. Despite the few recent efforts for detecting outliers [Xu

et al., 2007; Chakrabarti, 2004], which are vertices that do not belong to any dense subgraph, it is a general assumption that such cases are exceptions. However, our objective is to measure how attribute sets induce dense subgraphs and it is expected that in real graphs some attribute sets induce graphs for which the probability of a vertex to be in a dense subgraph is low.

In the remaining of this section we give several examples of dense subgraph definitions that could be applied in the correlation between attribute sets and dense subgraph formation. The simplest and most popular of them is a *clique* [Scott, 2000]. A set of vertices V is a clique if there is an edge between each pair of vertices in V . Cliques of interest are usually maximal subgraphs (i.e., they are not subsets of any other clique). Vertices can be naturally members of multiple cliques. However, the clique definition is too strict for real networks, which are known to be sparse [Newman, 2003]. As a consequence, large cliques are expected to be infrequent in real graphs.

There are several relaxations of the clique definition in the literature. A *quasi-clique* (see Section 2.2) is a set of vertices V such that each vertex is adjacent to a fraction γ of the vertices in V . The *k-plex* and *k-core* definitions are similar to quasi-cliques, but consider the absolute number of vertices adjacent to each vertex in the subgraph [Seidman and Foster, 1978; Seidman, 1983b]. A *k-plex* is a maximal vertex set V in which each vertex is adjacent to, at least, $|V| - k$ vertices in V . A *k-core* is a maximal vertex set V such that each vertex is adjacent to, at least, k vertices in V . There are also dense subgraph definitions based on the distances between vertices, such as *n-cliques* [Alba, 1973]. An *n-clique* is a maximal vertex set such that the distance between its vertices is not larger than n . Similar to cliques, dense subgraphs of interest are maximal in general.

Although most of the dense subgraph definitions are based only on internal cohesion, there are definitions that also consider the external cohesion. An example is the *LS-set* definition [Seidman, 1983a], which is a vertex set V such that the internal degree of each vertex in V is greater than its external degree. Hu et al. [Hu et al., 2008] define as *strong-community* a vertex set V for which the internal degree of any vertex in V exceeds its degree in any other vertex set. The same authors define a *weak-community* as a vertex set V such that its total internal degree exceeds the number of edges shared between the vertices in V and the other vertex sets. Borgatti et al. [Borgatti Martin and Stephen, 1990] consider the robustness to edge removal as a criterion for the identification of dense subgraphs through the definition of *lambda-sets*. A *lambda-set* is a vertex set V such that the connectivity between any pair of vertices in V is more robust than the connectivity between a vertex in V and another vertex

outside V .

Dense subgraphs can also be identified based on fitness measures that evaluate how a subgraph satisfies some cohesion criteria. An example of a fitness measure for dense subgraph identification is the *intra-cluster density*, which is the ratio between the number of internal edges and the number of all possible internal edges in the subgraph. A similar measure is the *relative density* of a vertex set V , which is the ratio between the internal and the total degree of V [Fortunato, 2010].

Some community definitions that consider overlapping communities and allow vertices not to be members of any community can also be applied in correlation between attribute sets and dense subgraph formation. The clique percolation method [Palla et al., 2005], for example, identifies overlapping communities through the union of k -cliques (i.e., cliques of size k) that share $k - 1$ vertices. Although the method requires the identification of the k -cliques, which is a known NP-complete problem, the limited number of cliques in real graphs makes the method applicable in many scenarios.

Identifying dense subgraphs in graphs is an important and popular research problem. However, we could not find a single well accepted definition of a dense subgraph in the literature. In the next section, we describe the quasi-clique mining task. Quasi-cliques are the specific definition of a dense subgraph used in this work. Nevertheless, other subgraph definitions can also be applied in the correlation between attribute sets and the formation of dense subgraphs.

2.2 Quasi-clique Mining

Quasi-cliques are a natural extension of the traditional clique definition. They fulfill the two requirements discussed in Section 2.1. In this work, we apply quasi-cliques as definition for dense subgraphs.

DEFINITION 1. (*Quasi-clique*) *Given a minimum density threshold γ_{min} ($0 < \gamma_{min} \leq 1$) and a minimum size threshold min_size , a quasi-clique is a maximal vertex set V such that for each $v \in V$, the degree of v in V is, at least, $\lceil \gamma_{min} \cdot (|V| - 1) \rceil$ and $|V| \geq min_size$.*

Figures 1.3 and 1.4 are examples of a 1-quasi-clique (i.e., a clique) of size 4 and a 0.6-quasi-clique of size 6, respectively, from the graph shown in Figure 1.1. The subgraph shown in Figure 1.5 is a 0.6-quasi-clique from the collaboration graph from Figure 1.2. The *quasi-clique mining problem* consists of identifying the set of quasi-cliques from a graph considering minimum size and density parameters.

DEFINITION 2. (*Quasi-clique mining problem*) Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where \mathcal{V} is the set of vertices and \mathcal{E} is the set of edges, a minimum density threshold γ_{min} ($0 < \gamma_{min} \leq 1$) and a minimum size threshold min_size . The problem consists of finding the set \mathcal{Q} of quasi-cliques from \mathcal{G} .

Quasi-cliques have shown great applicability as a definition for dense subgraphs in the data mining literature. In [Pei et al., 2005], the authors introduce the problem of mining cross-graph quasi-cliques (i.e., set of vertices that are quasi-cliques in every graph from a graph database). Cross-graph quasi-clique mining is useful for cross-market customer segmentation, correlated stock discovery, joint mining of gene expression and protein interactions, and the analysis of telecommunication data [Zeng et al., 2006; Pei et al., 2005; Abello et al., 2002; Jiang and Pei, 2009; Hu et al., 2005; Abello et al., 2002]. However, it is known that counting the number of quasi-cliques from a graph is a #P-hard problem and enumerating such quasi-cliques is an NP-hard problem, what is a challenge to the application of quasi-clique mining algorithms to large datasets. The class of #P-hard problems is the counting analogue of the class of NP-hard decision problems [Burgisser et al., 1997; Valiant, 1979; Garey and Johnson, 1990; Yang, 2004].

THEOREM 1. (*Quasi-clique mining problem complexity*). The problem of counting the number of γ -quasi-cliques in a graph \mathcal{G} is #P-hard and the problem of enumerating the γ -quasi-cliques from \mathcal{G} is NP-hard.

Proof sketch. As in Pei et al. [2005], we prove it by restriction. If γ is set to 1, the problem of counting the number of quasi-cliques becomes equivalent to the problem of counting the number of cliques (1-quasi-cliques) in \mathcal{G} , that is known to be #P-hard. #P-hard counting problems are associated to NP-hard enumeration problems [Yang, 2004; Garey and Johnson, 1990].

Figure 2.1 shows the search space for quasi-cliques considering a graph with 4 vertices (1-4) through an *enumeration tree* representation. Each node of the enumeration tree corresponds to a subset of vertices. According to the Theorem 1, a quasi-clique mining algorithm will enumerate every possible subset of the vertex set (i.e., the whole enumeration tree) in the worst case scenario. For a graph with N vertices, the height of the tree is N , the number of nodes in the level i ($0 \leq i \leq N$) is $\binom{N}{i}$, and the total number of nodes is 2^N .

There exist several efficient data mining algorithms for computationally hard problems in the literature. Such algorithms apply effective pruning techniques in order to eliminate candidate patterns as many and as soon as possible. Many of these pruning

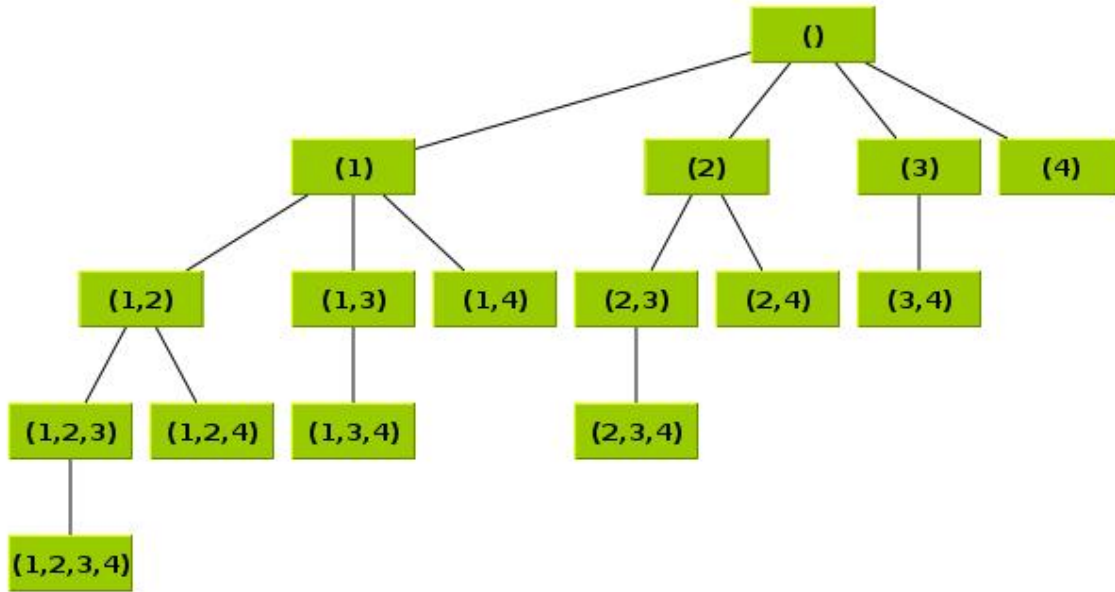


Figure 2.1: Set Enumeration Tree

techniques are based on the *anti-monotonicity* (or downward-closure) property [Manila and Toivonen, 1997; Ng et al., 1998] of patterns. However, the anti-monotonicity property does not hold for quasi-cliques.

DEFINITION 3. (*Anti-monotonicity or downward-closure property*). A property φ is called *anti-monotone* if and only if for any patterns P_1 and P_2 , the fact that $\varphi(P_1)$ holds implies that $\varphi(P_2)$ holds if $P_2 \subseteq P_1$.

PROPOSITION 1. (*Anti-monotonicity or downward-closure property of quasi-cliques*). The anti-monotonicity (or downward-closure) property does not hold for quasi-cliques.

Proof sketch. We prove it by a counterexample. The subgraph shown in Figure 1.4 is a 0.6-quasi-clique but its subgraph shown in Figure 2.2 is not a 0.6-quasi-clique, since the vertex 7 is connected to only one vertex in the subgraph.

Since quasi-cliques do not have the anti-monotonicity property, quasi-clique mining algorithms require more elaborated pruning strategies for the efficient enumeration of quasi-cliques. In the remaining of this section, we describe pruning techniques for quasi-clique mining proposed in the literature. Such pruning techniques are based on the quasi-clique definition and the input parameters (γ_{min} and min_size).

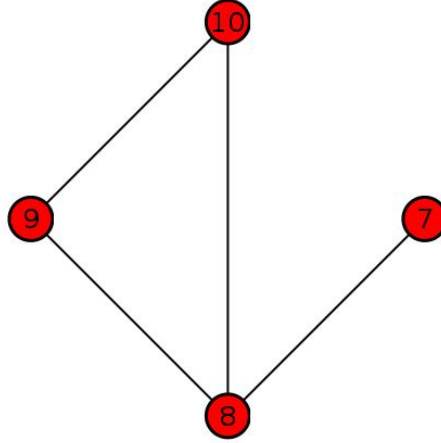


Figure 2.2: Subgraph of the 0.6-quasi-clique shown in Figure 1.3 which is not a 0.6-quasi-clique

2.2.1 Vertex Pruning

The vertex pruning techniques refer to the removal of vertices that can not be in any quasi-clique in a graph G according to the quasi-clique definition and the input parameters. Previous work [Pei et al., 2005; Jiang and Pei, 2009] has defined vertex pruning strategies based on the degree of vertices and the diameter of quasi-cliques, as stated in the following lemmas. Such techniques have as their main objective to reduce the number of vertices to be combined in the search for quasi-cliques.

LEMMA 1. (*Degree-based pruning*). *If the degree of a vertex v is smaller than $\lceil \gamma_{min} \cdot (min_size - 1) \rceil$, it can not be part of γ_{min} -quasi-clique of size greater or equal to min_size [Pei et al., 2005; Jiang and Pei, 2009].*

Based on Lemma 1, vertices can be pruned iteratively until no vertex can be removed. If $\gamma_{min} = 0.5$ and $min_size = 4$, vertices 1 and 2 can be pruned from the graph shown in Figure 1.1 using Lemma 1, since their degrees are smaller than 2.

LEMMA 2. (*Diameter of quasi-cliques*). *The maximum diameter ($diam(\gamma, \alpha)$) of a γ -quasi-clique of size α is limited by an upper-bound based on the values of γ_{min} and α [Pei et al., 2005; Jiang and Pei, 2009]:*

$$diam(\gamma_{min}, \alpha) \begin{cases} \leq 2 & \text{if } \frac{\alpha-2}{\alpha-1} \geq \gamma_{min} \geq 0.5 \\ = 1 & \text{if } 1 \geq \gamma_{min} > \frac{\alpha-2}{\alpha-1} \end{cases}$$

Lemma 2 can be applied to prune vertices which do not have enough neighbors within a shortest path smaller or equal to the upper bound of the quasi-clique diameter.

LEMMA 3. (*Diameter-based pruning*). *Let $d(u, v)$ be the number of edges in the shortest path between u and v , and $N^k(v)$ be the size k neighborhood of a vertex v , i.e., $N^k(v) = \{u | d(u, v) \leq k\}$. If $|N^{\text{diam}(\gamma_{\min}, \text{min_size})}(v)| < \text{min_size} - 1$, v can not be in any γ_{\min} -quasi-clique of size greater or equal to min_size [Pei et al., 2005; Jiang and Pei, 2009].*

According to Lemma 3, if $\gamma_{\min} = 0.5$ and $\text{min_size} = 6$, then vertex 4 can be removed from the graph from Figure 1.1, since $\text{diam}(0.5, 6) \leq 2$ and $|N^2(4)| < 5$. Similar to Lemma 1, Lemma 3 can be applied iteratively in order to prune as many vertices as possible. Lemma 1 and Lemma 3 can also be combined in order to maximize the number of vertices removed. If $\gamma_{\min} = 0.7$ and $\text{min_size} = 4$, vertices 1 and 2 can be pruned from the graph shown in Figure 1.1 using Lemma 3, since the upper bound for the diameter of size 4 0.7-quasi-clique is 1, and $|N^1(1)|$ and $|N^1(2)|$ are smaller than 3.

In the next section, we describe a second group of pruning techniques for quasi-clique mining, which we call *candidate quasi-clique pruning*. While the vertex pruning techniques remove vertices that can not be in any quasi-clique, the candidate quasi-clique pruning removes sets of vertices in the search space for quasi-cliques.

2.2.2 Candidate Quasi-clique Pruning

Candidate quasi-clique pruning techniques were proposed by previous work [Pei et al., 2005; Jiang and Pei, 2009; Zeng et al., 2006; Liu and Wong, 2008] in order to speedup the quasi-clique mining process. Based on the quasi-clique definition and the input parameters, such techniques are able to prune an entire branch from the set enumeration tree that represents the search space for quasi-cliques (see Figure 2.1). The set of possible candidates to be quasi-cliques C (i.e., nodes of the corresponding enumeration tree) from a vertex set \mathcal{V} can be generated through the Algorithm 1. It is based on two vertex sets: X and $\text{candExts}(X)$. The set X is the current vertex set, initially set to \emptyset , and the set $\text{candExts}(X)$ stores vertices to extend X , initially set as the vertex set \mathcal{V} . The function **extend** expands X recursively using vertices from $\text{candExts}(X)$, one at time. In order to avoid duplicated subsets, the set of new candidate extensions is limited to those vertices greater than the new vertex inserted into X according to some comparison criteria. The pruning techniques for quasi-clique discovery will be described in terms of the sets X and $\text{candExts}(X)$, and the parameters γ_{\min} and min_size . Moreover, we define as $N^G(v)$ the set of vertices adjacent to v in the input

graph. The function $indeg^X$ gives the degree of a vertex in X , i.e. $indeg^X(v) = |N^G(v) \cap X|$, and $exdeg^X$ gives the degree of a vertex in $candExts(X)$, i.e. $exdeg^X(v) = |N^G(v) \cap candExts(X)|$.

Algorithm 1 Set Enumeration Tree Algorithm

INPUT: V : Vertex set

OUTPUT: C : Power set of \mathcal{V}

1: $X \leftarrow \emptyset$

2: $candExts(X) \leftarrow \mathcal{V}$

3: $C \leftarrow X$

4: $C \leftarrow C \cup \text{extend}(X, candExts(X))$

Algorithm 2 extend

INPUT: $X, candExts(X)$

OUTPUT: E

1: $E \leftarrow \emptyset$

2: **for all** $v \in candExts(X)$ **do**

3: $candExts(newX) \leftarrow \{u \in candExts(X) \mid u > v\}$

4: $newX \leftarrow X \cup \{v\}$

5: $E \leftarrow E \cup newX$

6: $E \leftarrow E \cup \text{extend}(newX, candExts(newX))$

7: **end for**

The first two candidate quasi-clique pruning techniques consider the degree of vertices in $X \cup candExts(X)$ in order to prune combinations of vertices that can not be quasi-cliques. The third technique prunes vertices from $candExts(X)$ that are not reachable by the vertices from X according to the diameter upper-bound of a quasi-clique, described in Lemma 2.

LEMMA 4. (Degree-based pruning for $candExts$). For a pair $(X, candExts(X))$, if $indeg^X(u) + exdeg^X(u) < \lceil \gamma_{min} \cdot (|X| + exdeg^X(u)) \rceil$, for a given vertex $u \in candExts(X)$, u can be pruned from $candExts(X)$ [Zeng et al., 2006].

Considering the example graph shown in Figure 1.1, if $X = \{3, 4, 5\}$, $candExts(X) = \{7\}$, and $\gamma_{min} = 0.5$, the vertex 7 can be pruned from $candExts(X)$, since $indeg^X(7) = 1$, $exdeg^X(7) = 0$, and $indeg^X(7) + exdeg^X(7) < \lceil 0.5 \cdot (4 + exdeg^X(7)) \rceil$.

LEMMA 5. (Degree-based pruning for X). For a pair $(X, candExts(X))$, if $indeg^X(v) < \lceil \gamma_{min} \cdot |X| \rceil$ and $exdeg^X(v) = 0$, or $indeg^X(v) + exdeg^X(v) < \lceil \gamma_{min} \cdot (|X| - 1 + exdeg^X(v)) \rceil$, for a given a vertex $u \in X$, v can be pruned from X [Zeng et al., 2006].

Lets consider the graph from Figure 1.1, $X = \{4, 6, 7, 8, 9\}$, $candExts(X) = \{10, 11\}$, and $\gamma_{min} = 0.5$. According to Lemma 5, the vertex 4 can be pruned from

X , since $\text{indeg}^X(3) = 1$, $\text{exdeg}^X(3) = 0$, and $\text{indeg}^X(4) + \text{exdeg}^X(4) < \lceil 0.5 \cdot (5 - 1 + \text{exdeg}^X(4)) \rceil$.

LEMMA 6. (*Diameter-based pruning*). *Let $d(u, v)$ be the number of edges in the shortest path between u and v , and $N^k(v)$ be the size k neighborhood of a vertex v , i.e., $N^k(v) = \{u \mid d(u, v) \leq k\}$. For a pair $(X, \text{candExts}(X))$, if $u \notin \bigcap_{v \in X} N^{\text{diam}(\gamma_{\min}, \text{min-size})}(v)$, for a given $u \in \text{candExts}(X)$, u can be pruned from $\text{candExts}(X)$ [Pei et al., 2005; Jiang and Pei, 2009].*

As an example, let $X = \{3, 4, 5, 6\}$, $\text{candExts}(X) = \{10, 11\}$, $\gamma_{\min} = 0.5$, and the input graph to be the graph from Figure 1.1. The vertex 11 can be pruned from $\text{candExts}(X)$ based on Lemma 6, since $11 \notin N^2(3) \cap N^2(4) \cap N^2(5) \cap N^2(6)$. The following 4 lemmas, proposed in [Liu and Wong, 2008], consider the upper bound of the number of vertices that can be added to X to generate larger quasi-cliques in order to prune quasi-clique candidates.

LEMMA 7. (*Upper bound of the number of vertices from candExts that can be added to X*). *Let $\text{deg}_{\min}(X) = \min\{\text{indeg}^X(v) + \text{exdeg}^X(v) \mid v \in X\}$. For a pair $(X, \text{candExts}(X))$, an upper bound of the number of vertices from $\text{candExts}(X)$ which can be included in X , U_X^{\min} , to form a γ_{\min} -quasi-clique is given by $\lfloor \text{deg}_{\min}(X) / \gamma_{\min} \rfloor + 1 - |X|$.*

LEMMA 8. (*Tighter upper bound of the number of vertices from candExts that can be added to X*). *A tighter upper bound of the number of vertices from $\text{candExts}(X)$ that can be added to X , U_X , to form a γ_{\min} -quasi-clique is given by $\max\{t \mid \sum_{v \in X} \text{indeg}^X(v) + \sum_{1 \leq i \leq t} \text{indeg}^X(v_i) \geq |X| \cdot \lceil \gamma_{\min} \cdot (|X| + t - 1) \rceil, 1 \leq t \leq U_X^{\min}, v_i \in \text{candExts}\}$. If there is no such t , U_X is set to 0.*

LEMMA 9. (*Pruning for candExts based on the upper bound of the number of vertices from candExts that can be added to X*). *For a pair $(X, \text{candExts}(X))$, if $U_X = 0$, then the entire set $\text{candExts}(X)$ can be pruned, otherwise, if $\text{indeg}^X(u) + U_X - 1 < \lceil \gamma_{\min} \cdot (|X| + U_X - 1) \rceil$, for a given vertex $u \in \text{candExts}(X)$, then u can be pruned from $\text{candExts}(X)$.*

Back to our example graph shown in Figure 1.1, if $X = \{6, 7, 8\}$, $\text{candExts} = \{9, 10, 11\}$, and $\gamma_{\min} = 0.9$, then $\text{deg}_{\min}(X) = 3$ and $U_X^{\min} = 1$. If we apply Lemma 8, then $U_X = 0$ and, according to Lemma 9, the set X can not be extended to generate a larger 0.9-quasi-clique.

LEMMA 10. (*Pruning for X based on the upper bound of the number of vertices from candExts that can be added to X*). *For a pair $(X, \text{candExts}(X))$,*

if $\text{indeg}^X(v) + U_X < \lceil \gamma_{\min} \cdot (|X| + U_X - 1) \rceil$, for a given vertex $v \in X$, then v can be pruned from X .

In order to illustrate the application of Lemma 10, let $X = \{1, 4, 7\}$, $\text{candExts}(X) = \{8, 11\}$, $\gamma_{\min} = 0.5$, and the input graph from Figure 1.1. According to Lemmas 7 and 8, $U_X^{\min} = U_X = 0$. Therefore, the vertex 7 can be pruned based on Lemma 10, since $\text{indeg}^X(7) = 0$ and $0 + 0 < \lceil 0.5(3 + 0 - 1) \rceil$.

Liu and Wong [Liu and Wong, 2008] also proposed pruning rules based on the lower bound of the number of vertices from $\text{candExts}(X)$ to be added to X , as stated by the following 5 lemmas.

LEMMA 11. (Lower bound of the number of vertices from candExts that can be added to X). Let $\text{indeg}_{\min}(X) = \min\{\text{indeg}^X(v) | v \in X\}$. A lower bound L_X^{\min} for the number of vertices from $\text{candExts}(X)$ that can be added to X to form a γ_{\min} -quasi-clique is given by $\min\{t | \text{indeg}_{\min}(X) + t \geq \lceil \gamma_{\min} \cdot (|X| + t - 1) \rceil\}$.

LEMMA 12. (Tighter lower bound of the number of vertices from candExts that can be added to X). A tighter lower bound of the number of vertices from $\text{candExts}(X)$ that can be added to X , L_X , to form a γ_{\min} -quasi-clique is given by $\min\{t | \sum_{v \in X} \text{indeg}^X(v) + \sum_{1 \leq i \leq t} \text{indeg}^X(v_i) \geq |X| \cdot \lceil \gamma_{\min} \cdot (|X| + t - 1) \rceil, L_X^{\min} \leq t \leq |\text{candExts}(X)|\}$. If there is no such t , $L_X = |\text{candExts}(X)| + 1$.

LEMMA 13. (Pruning for candExts based on the lower bound of the number of vertices from candExts that can be added to X). For a pair $(X, \text{candExts}(X))$, if $L_X > U_X$ the entire set $\text{candExts}(X)$ can be pruned, otherwise, if $\text{indeg}^X(u) + \text{exdeg}^X(u) < \lceil \gamma_{\min} \cdot (|X| + L_X - 1) \rceil$, for a given vertex $u \in \text{candExts}(X)$, u can be pruned from $\text{candExts}(X)$.

Considering the graph from Figure 1.1, if $X = \{1, 4, 7\}$, $\text{candExts}(X) = \{8, 11\}$, and $\gamma_{\min} = 0.5$, then $L_X^{\min} = 2$ and $L_X = 3$, according to Lemmas 11 and 12, respectively. Moreover, $U_X = 0$, according to Lemma 8. Therefore, $U_X > L_X$ and, based on Lemma 13, X can not be extended to generate a larger quasi-clique.

LEMMA 14. (Pruning for X based on the lower bound of the number of vertices from candExts that can be added to X). For a pair $(X, \text{candExts}(X))$, if $\text{indeg}^X(v) + \text{exdeg}^X(v) < \lceil \gamma_{\min} \cdot (|X| + L_X - 1) \rceil$, for a given vertex $v \in X$, v can be pruned from X .

Let $X = \{2, 3, 8\}$, $\text{candExts}(X) = \{9, 10, 11\}$, $\gamma_{\min} = 0.5$, and the graph from Figure 1.1. The vertices 2 and 3 can be pruned from X based on Lemma 14, since $L_X = 4$, $\text{indeg}^X(2) + \text{exdeg}^X(2) = \text{indeg}^X(3) + \text{exdeg}^X(3) = 1$, and $1 < \lceil 0.5 \cdot (3 + 4 - 1) \rceil$.

LEMMA 15. (*Pruning based on critical vertices*). *A vertex $v \in X$ is called critical if $\text{indeg}^X(v) + \text{exdeg}^X(v) = \lceil \gamma_{\min} \cdot (|X| + L_X - 1) \rceil$. For each critical vertex v , we add every vertex $u \in \text{candExts}(X)$, such that $(u, v) \in E$ (i.e., vertices in $\text{candExts}(X)$ that are adjacent to v), to X [Liu and Wong, 2008].*

Let $X = \{6, 7, 8, 9\}$, $\text{candExts}(X) = \{10, 11\}$, $\gamma_{\min} = 0.6$, and the graph from Figure 1.1. According to 15, the vertex 9 is a critical vertex because $L_X = 1$ and $\text{indeg}^X(9) + \text{exdeg}^X(9) = \lceil 0.6(4 + 1 - 1) \rceil = 3$.

Liu and Wong [Liu and Wong, 2008] also generalize a technique for mining maximal cliques [Tomita et al., 2006] to prune non-maximal quasi-cliques. Such technique is based on the concept of cover vertex, which is a vertex u from $\text{candExts}(X)$ that prevents the vertices covered by it to form a maximal quasi-clique without u .

LEMMA 16. (*Pruning based on cover vertices*). *For a pair $(X, \text{candExts}(X))$, let $u \in \text{candExts}(X)$ be a vertex such that $\text{indeg}^X(u) \geq \lceil \gamma_{\min} \cdot |X| \rceil$. If $\text{indeg}^X(v) \geq \lceil \gamma_{\min} \cdot |X| \rceil$ for any vertex $v \in X$ such that $(u, v) \notin E$ (i.e., u is not adjacent to v), then vertices in $\text{candExts}(X) \cap N^G(u) \cap (\bigcap_{v \in X \wedge (u,v) \notin E} N^G)$ must be extended together with u in order to form a maximal γ_{\min} -quasi-clique. Any extension of X that includes $\text{candExts}(X) \cap N^G(u) \cap (\bigcap_{v \in X \wedge (u,v) \notin E} N^G)$ but not u can be pruned.*

Considering our example graph 1.1, let $X = \{6, 7, 8\}$, $\text{candExts} = \{9, 10, 11\}$, and $\gamma_{\min} = 0.5$. Based on Lemma 16, the vertex 9 covers the vertex 11, since $\text{indeg}^X(9) = 2 = \lceil 0.5 \cdot (3) \rceil$, and $\text{candExts}(X) \cap N^G(9) \cap N^G(7) = \{11\}$. Therefore, $\{6, 7, 8, 11\}$ can be pruned as it can not be a maximal 0.5-quasi-clique. According to Lemma 16, the cover vertex with the largest covering can be identified in order to prune as many extensions from X as possible.

LEMMA 17. (*Lookahead pruning*). *Since quasi-cliques of interest are usually maximal, for a pair $(X, \text{candExts}(X))$, the set $X \cup \text{candExts}(X)$ can be checked before extending X . If $X \cup \text{candExts}(X)$ is a γ_{\min} -quasi-clique, all the extensions of X can be pruned, since they can not be larger than $|X \cup \text{candExts}(X)|$ [Liu and Wong, 2008].*

The lookahead pruning technique is very simple but can avoid the checking of several vertex combinations in the cases where $X \cup \text{candExts}(X)$ is a quasi-clique. Let $X = \{6, 7, 8\}$, $\text{candExts}(X) = \{9, 10, 11\}$, $\gamma_{\min} = 0.5$, and the input graph be the example graph shown in Figure 1.1. We know that $\{6, 7, 8, 9, 10, 11\}$ is a 0.5-quasi-clique (see Figure 1.4) and thus its subsets can not be maximal quasi-cliques.

In Chapter 4, we present algorithms for the structural correlation pattern mining problem. Such algorithms apply the pruning techniques described along this section in

the identification of dense subgraphs induced by attribute sets. Moreover, we propose search, pruning, sampling, and parallelization techniques for correlating attribute sets and dense subgraphs efficiently.

2.3 Frequent Itemset Mining

In this section we discuss the frequent itemset mining problem [Agrawal et al., 1993; Agrawal and Srikant, 1994; Ceglar and Roddick, 2006; Hipp et al., 2000], which is an important research topic related to the problem of measuring the correlation between attributes and the formation of dense subgraphs. Mining frequent itemsets is one of the most traditional problems in data mining. Frequent itemset mining algorithms have been integrated to several other algorithms in order to find frequent sets of items that can be used to express a reduced set of relevant patterns. Similarly to the frequent sequence mining and the frequent subgraph mining problems, the frequent itemset mining is considered part of the class of frequent pattern mining problems.

Agrawal et al. introduced the frequent itemset mining problem in [Agrawal et al., 1993]. It consists of identifying frequent itemsets in a transactional database (each transaction is a set of items) according to a minimum support threshold.

DEFINITION 4. (*Frequent itemset mining problem*) Given a set of items \mathcal{I} , a database $\mathcal{D} = \langle T_1, T_2, \dots, T_n \rangle$, where $T_i \subseteq \mathcal{I}$ ($0 \leq i < n$), a support function such that $\text{support}(X) = |\{T \in \mathcal{D} | X \subseteq T\}|$, and a user-defined minimum support threshold min_sup . The problem consists of identifying the set of frequent itemsets \mathcal{F} , such that $\mathcal{F} = \{X \subseteq \mathcal{I} | \text{support}(X) \geq \text{min_sup}\}$.

The main challenge for frequent itemset mining algorithms is the efficient enumeration of frequent itemsets from large databases. Similarly to the quasi-clique mining, counting the number of frequent itemsets is a #P-hard problem and enumerating such itemsets is an NP-hard problem.

THEOREM 2. (*Frequent itemset mining problem complexity*). Counting the number of frequent itemsets from a database is a #P-hard problem and enumerating the frequent itemsets is an NP-hard problem.

Proof sketch. The problem of computing the number of possible assignments of a monotone-2CNF formula, which is #P-hard, can be reduced to the problem of mining frequent itemsets in polynomial time, as shown in [Gunopulos et al., 2003]. The NP-hardness of the associated enumeration problem can be directly derived [Yang, 2004].

The enumeration of the frequent itemsets requires, in the worst case, a search over the $2^{|\mathcal{I}|} - 1$ possible combinations of items from the database, where \mathcal{I} is the set of items. Figure 2.3 shows such a search space for the set of items $\{A, B, C, D, E\}$ in the form of a lattice structure, where edges represent subset/superset relationships between itemsets. Table 2.1 shows the list of frequent itemsets, with their respective supports, from the vertex attributes presented in Table 1.1 for a minimum support set to 3. Different from quasi-cliques, the anti-monotonicity property holds for frequent itemsets. Frequent itemset mining algorithms exploit such property in order to prune candidate itemsets in an incremental generation process.

PROPOSITION 2. (*Anti-monotonicity or downward-closure property of frequent itemsets*). *The anti-monotonicity (or downward-closure) property holds for quasi-cliques.*

Proof sketch. *For any transaction T_i from the database, if an itemset I is contained in T_i , then any subset of I is also contained in T_i . Therefore, the support of any subset of I is, at least, the support of I .*

attribute set	support
A	11
B	6
C	3
D	3
A, B	6
A, C	3

Table 2.1: Frequent attribute sets from the vertex attributes shown in Table 1.1 (the minimum support set is 3)

The identification of frequent itemsets has several applications. The motivational problem for the frequent itemset mining problem was the extraction of association rules in market transaction data [Agrawal et al., 1993; Agrawal and Srikant, 1994]. Moreover, frequent itemsets have been applied in clustering [Zhang et al., 2010], classification [Thabtah, 2007], and information retrieval [Pôssas et al., 2005]. The original problem of mining frequent itemsets was further extended to the problem of mining high utility itemsets, where utility values are associated with items in the database [Chan et al., 2003].

There are several algorithms for frequent itemset mining in the literature [Ceglar and Roddick, 2006; Hipp et al., 2000]. Such algorithms exploit different search strategies, pruning techniques, data structures and dataset organizations in order to identify

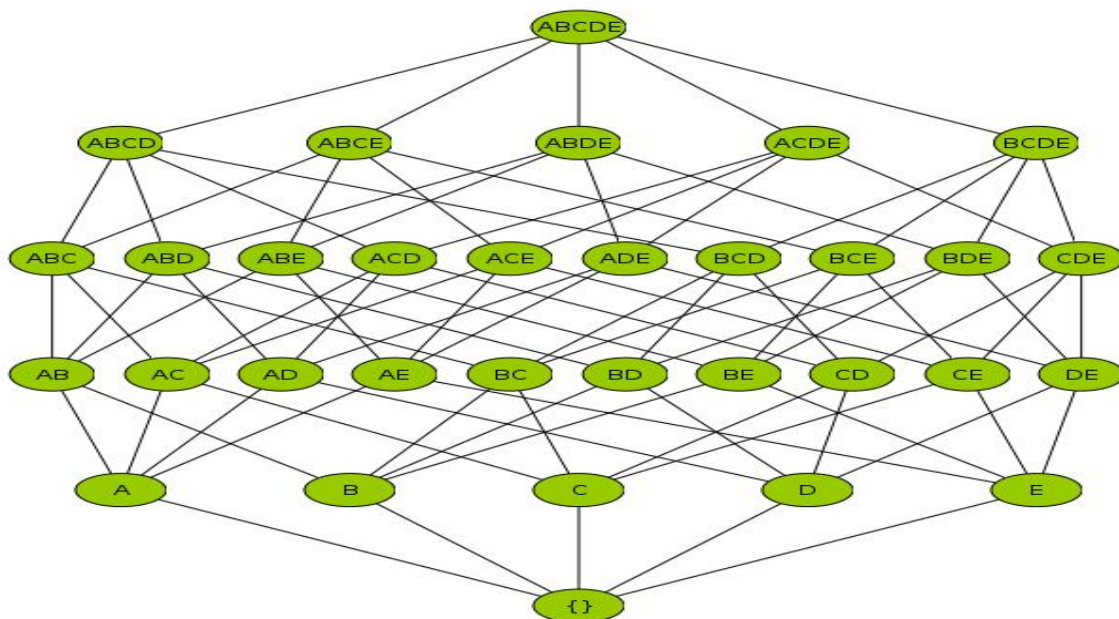


Figure 2.3: Complete lattice for the set of attributes $\{A, B, C, D, E\}$

frequent itemsets efficiently. For a survey on algorithms for frequent itemset mining see [Ceglar and Roddick, 2006]. A comparison among several existing frequent itemset mining algorithms is presented in [Hipp et al., 2000].

In this work, we apply frequent itemset mining in order to select attribute sets to be evaluated in terms of their capacity of inducing dense subgraphs in an attribute graph. In this case, each attribute is an item and an attribute set is frequent if its corresponding itemset is frequent. Analyzing all possible attribute sets becomes computationally infeasible if the number of attributes is large. Moreover, infrequent attribute set may not be of interest because: (1) they cover a small part of the graph and (2) the evidence about their correlation with the formation of dense subgraphs may not be enough to identify a relevant pattern.

2.4 Related Work

In this section, we discuss some work related to the structural correlation pattern mining from the literature. Finding communities [Girvan and Newman, 2002; Fortunato, 2010] and dense subgraphs [Gibson et al., 2005; Liu and Wong, 2008; Jiang and Pei, 2009; Zeng et al., 2006] has been a long term research topic. A community is usually defined as set of vertices significantly more connected among themselves than with vertices outside it [Fortunato, 2010]. On the other hand, dense subgraph definitions,

such as cliques, are strongly based on internal cohesion and maximality.

This work applies a specific dense subgraph definition called quasi-clique (see Section 2.2). [Pei et al., 2005] introduces the problem of mining cross-graph quasi-cliques. They further studied the problem of mining frequent cross-graph quasi-cliques [Jiang and Pei, 2009]. [Zeng et al., 2006] studies the problem of mining frequent coherent closed quasi-cliques. [Liu and Wong, 2008] studies the problem of finding the set of quasi-cliques from a single graph, proposing several powerful pruning techniques for quasi-clique mining. We apply the same pruning techniques described in [Liu and Wong, 2008].

Traditional graph community and dense subgraph definitions are based on the topology of graphs (i.e., vertices and edges). However, in several scenarios, it is expected that vertex properties may complement or be associated to the graph topology. In the specific case of social networks, important phenomena such as homophily [McPherson et al., 2001] and social influence [Anagnostopoulos et al., 2008] result in a significative similarity between connected nodes. The concept of social correlation, defined in [Anagnostopoulos et al., 2008], which is the co-occurrence of a particular event for two adjacent nodes, motivated the study of the structural correlation in graphs.

Graph clustering and dense subgraph discovery methods that consider vertex attributes as complementary information have attracted the interest of the research community in the recent years [Moser et al., 2009; Ge et al., 2008; Zhou et al., 2009; Mougél et al., 2010]. [Ge et al., 2008] and [Zhou et al., 2009] propose algorithms for community detection based on the graph topology and vertex attributes. In [Moser et al., 2009], the authors introduce the problem of mining cohesive patterns, which are dense connected subgraphs where vertices have homogeneous attributes (or features). [Mougél et al., 2010] considers the problem of computing maximal homogeneous cliques in attributed graphs.

Combining structural and attribute information in graphs can be seen as a special case of *multi-relational data mining* [Džeroski, 2003; Wrobel, 2000]. This class of data mining problems deal with patterns that involve multiple relations (i.e., tables). Multi-relational data mining is strongly related to inductive logic programming and has interesting applications, specially in bioinformatics. In this paper, we are interested in the particular problem of correlating vertex attributes and dense subgraphs in attributed graphs.

Assessing how vertex attributes are related to the graph topology has led to the definition of innovative patterns. In [Silva et al., 2010], we introduce the concept of structural correlation as a measure of how attribute sets induce dense subgraphs in attributed graphs. [Khan et al., 2010] defines the proximity pattern mining, which

evaluates the proximity among attributes in a graph. [Sese et al., 2010] proposed the problem of finding itemset-sharing itemsets, which consists of extracting subgraphs with common itemsets. In [Guan et al., 2011], the authors propose a different definition for the structural correlation, which compares the closeness among vertices induced by a given graph against a subgraph where attributes are randomly distributed. This work differs from [Guan et al., 2011] by considering a particular topological property which is the organization into dense subgraphs. Moreover, we are interested in relevant dense subgraphs to be representatives of the structural correlation at the attribute level. Some of the main results of this thesis can be found in [Silva et al., 2012].

Along this chapter, we have discussed the main topics related to the structural correlation pattern mining from the literature: dense subgraphs and the quasi-clique and frequent itemset mining problems, which will be relevant for the understanding of the subsequent chapters. We have also described related work on structural correlation pattern mining from the literature. In the next chapter, we give formal definitions related to the structural correlation pattern mining.

Chapter 3

Structural Correlation Pattern Mining: Definitions

Correlating vertex attributes and the existence of dense subgraphs in large attributed graphs is the problem studied in this work. Dense subgraphs occur in several real graphs (e.g., social networks, biological networks) and attributes can be assigned to vertices in many of them. In this scenario, there are some interesting questions to ask, such as: What is the relationship between vertex attributes and the formation of dense subgraphs? Do vertex attributes induce dense subgraphs in real graphs? For a given graph, what are the attributes more related to the formation of dense subgraphs? And what are the dense subgraphs induced by these attributes? Along this chapter, we define new data mining problems in order to answer these questions.

Section 3.1 defines the *structural correlation*, which measures to what extent vertices with a given attribute set are organized into dense subgraphs. More specifically, the structural correlation of an attribute set is the probability of a vertex to be member of a dense subgraph in the graph induced by such attributes. Moreover, we also propose null models that give the expected structural correlation of an attribute set considering the input graph topology, the attribute set frequency, and the quasi-clique parameters. Based on such null models, Section 3.2 describes two normalization approaches for the structural correlation function. The normalized structural correlation gives how the structural correlation of an attribute set deviates from its expected value.

The concept of *structural correlation pattern* is defined in Section 3.3. A structural correlation pattern is a dense subgraph for which vertices share a specific attribute set. Therefore, while the structural correlation is an analysis at the level of attributes, structural correlation patterns provide an analysis at the level of dense subgraphs induced by attributes. As discussed in Section 2.2, we consider quasi-cliques as a

definition for dense subgraphs.

In Section 3.4, we put the concepts of structural correlation and structural correlation pattern together to define the *structural correlation pattern mining problem*, which is the identification of attribute sets with high structural correlation and the dense subgraphs induced by them.

3.1 Structural Correlation

In this section we give a formal definition for the concept of *structural correlation*. The structural correlation measures the probability of vertices to be in a quasi-clique in the graph induced by a given attribute set.

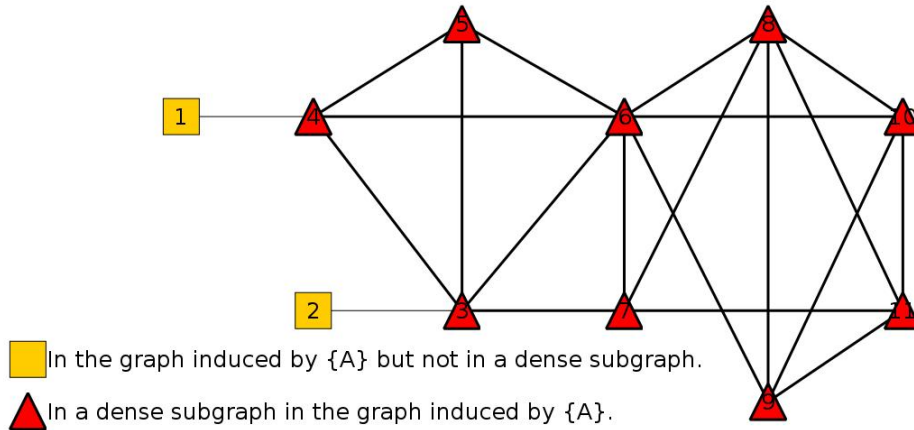
We define an attributed graph as a 4-tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{F})$ where \mathcal{V} is the set of vertices, \mathcal{E} is the set of edges, $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ is the set of vertex attributes, and $\mathcal{F} : \mathcal{V} \rightarrow P(\mathcal{A})$ is a function that returns the set of attributes of a vertex. P is the power set function. Each vertex v_i in \mathcal{V} has a set of attributes $\mathcal{F}(v_i) = \{a_{i1}, a_{i2}, \dots, a_{ip}\}$, where $p = |\mathcal{F}(v_i)|$ and $\mathcal{F}(v_i) \subseteq \mathcal{A}$. In Figure 1.1, we show an example of an attributed graph where the vertex attributes are given in Table 1.1. In such graph, $\mathcal{V} = \{1, 2, \dots, 11\}$, $\mathcal{A} = \{A, B, C, D, E\}$, $(1, 4) \in \mathcal{E}$, and $\mathcal{F}(1) = \{A, C\}$. In the collaboration graph shown in Figure 1.2, \mathcal{V} is a set of researchers, \mathcal{A} is composed of keywords, \mathcal{E} represents collaborations, and $\mathcal{F}(r_i)$ returns the set of keywords associated to the researcher r_i . Table 3.1 presents the set of symbols used along this and the upcoming chapters.

Given the set of vertex attributes \mathcal{A} , we define an *attribute set* S as a subset of \mathcal{A} ($S \subseteq \mathcal{A}$). Moreover, we denote by $\mathcal{V}(S) \subseteq \mathcal{V}$ the vertex set induced by S (i.e., $\mathcal{V}(S) = \{v_i \in \mathcal{V} | S \subseteq \mathcal{F}(v_i)\}$) and by $\mathcal{E}(S) \subseteq \mathcal{E}$ the edge set induced by S (i.e., $\mathcal{E}(S) = \{(v_i, v_j) \in \mathcal{E} | v_i \in \mathcal{V}(S) \wedge v_j \in \mathcal{V}(S)\}$). Therefore, the graph $\mathcal{G}(S)$, induced by S , is the pair $(\mathcal{V}(S), \mathcal{E}(S))$. Considering the graph shown in Figure 1.1, with vertex attributes given in Table 1.1, $\{A\}$, $\{C\}$, and $\{A, B\}$ are examples of attribute sets. In Figures 3.1, 3.2, and 3.3, we show different versions of the graph from Figure 1.1 in which vertices in the graph induced by an attribute set but not in dense subgraphs, vertices in the graph induced by an attribute set and in dense subgraphs, and vertices out of the induced graph are set to different shapes and colors, for the attribute sets $\{A\}$, $\{C\}$, and $\{A, B\}$, respectively. Figure 3.4 shows the graph induced by the keywords “search” and “rank” in the collaboration graph from Figure 1.2. For clarity, vertices that do not have the attribute set $\{search, rank\}$ are not shown.

Given the induced subgraph $\mathcal{G}(S)$, the structural correlation measures how its vertices are organized into dense subgraphs. We apply quasi-cliques (see Section 2.2)

symbol	meaning
\mathcal{G}	attributed graph
\mathcal{V}	vertex set of \mathcal{G}
\mathcal{E}	edge set of \mathcal{G}
\mathcal{A}	attribute set of \mathcal{G}
$\mathcal{F}(v)$	set of attributes of v in \mathcal{G}
S	attribute set
$\mathcal{V}(S)$	$\{v v \in \mathcal{V} \wedge S \subseteq \mathcal{F}(v)\}$
$\mathcal{E}(S)$	$\{(v, u) v \in \mathcal{V}(S) \wedge u \in \mathcal{V}(S)\}$
$\mathcal{G}(S)$	$(\mathcal{V}(S), \mathcal{E}(S))$
$\kappa(S)$	structural coverage of S
$\epsilon(S)$	structural correlation of S
$\sigma(S)$	$ \mathcal{V}(S) $
σ_{min}	minimum support threshold
$\text{sim-}\epsilon_{exp}$	simulation-based expected ϵ
$\text{max-}\epsilon_{exp}$	analytical upper bound on the expected ϵ
δ_1	simulation based normalized ϵ
δ_2	analytical normalized ϵ
γ_{min}	minimum density threshold for quasi-cliques
min_size	minimum size threshold for quasi-cliques

Table 3.1: Table of Symbols

Figure 3.1: Graph from Figure 1.1, vertices 3 to 11 have the attribute A and are in dense subgraphs, vertices 1 and 2 have the attribute A but are out of dense subgraphs

as a definition for dense subgraphs. The structural correlation is defined based on the *structural coverage function*, which measures how many vertices are covered by quasi-cliques in the graph induced by a given attribute set.

DEFINITION 5. (Structural coverage function κ) Given an attribute set S , let q_1, q_2, \dots, q_k be the set of quasi-cliques in the graph $\mathcal{G}(S)$. The value of $\kappa(S)$ is given by:

$$\kappa(S) = \left| \bigcup_{0 < i \leq k} q_i \right| \quad (3.1)$$

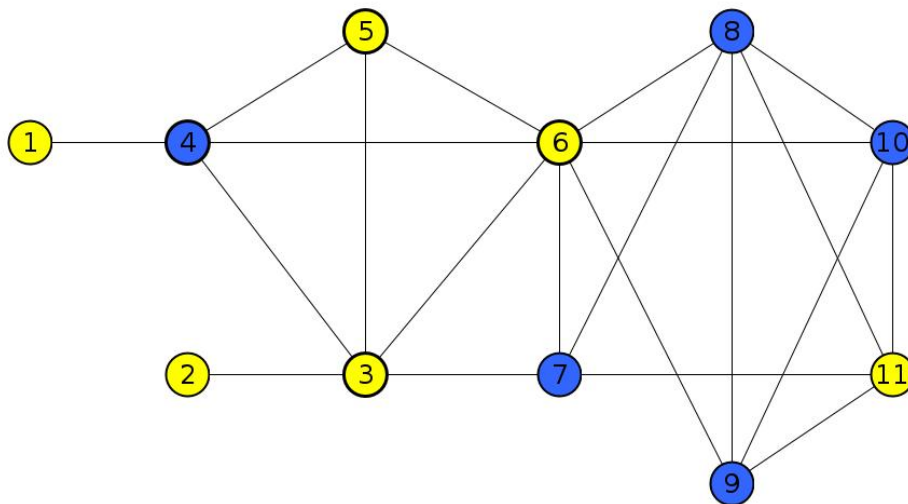


Figure 3.2: Graph from Figure 1.1, vertices 1, 3, and 6 have the attribute C and the other vertices do not have C

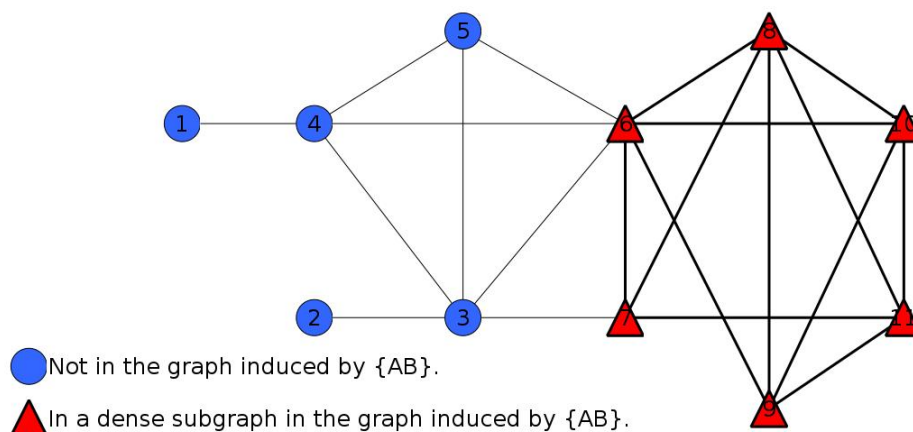


Figure 3.3: Graph from Figure 1.1, vertices 6 to 11 have the attribute set $\{A, B\}$ and are in dense subgraphs, the other vertices do not have $\{A, B\}$

In Figures 3.1, 3.2, 3.3, and 3.4, vertices covered by quasi-cliques in the induced subgraph and vertices in the induced subgraph but not covered by quasi-cliques are set to distinct shapes and colors. The value of $\kappa(\{A\})$, $\kappa(\{C\})$, and $\kappa(\{A, B\})$ are 9, 0, and 6, respectively. In the collaboration graph from Figure 3.4, $\kappa(\{search, rank\})$ is equal to 81 if γ and min_size are set to 0.5 and 10, respectively. We define the *structural correlation function* for an attribute set S as the probability of a vertex v that has S (i.e., $v \in \mathcal{V}(S)$) to be covered by a dense subgraph in $\mathcal{G}(S)$.

DEFINITION 6. (*Structural correlation function ϵ*) Given an attribute set S ,

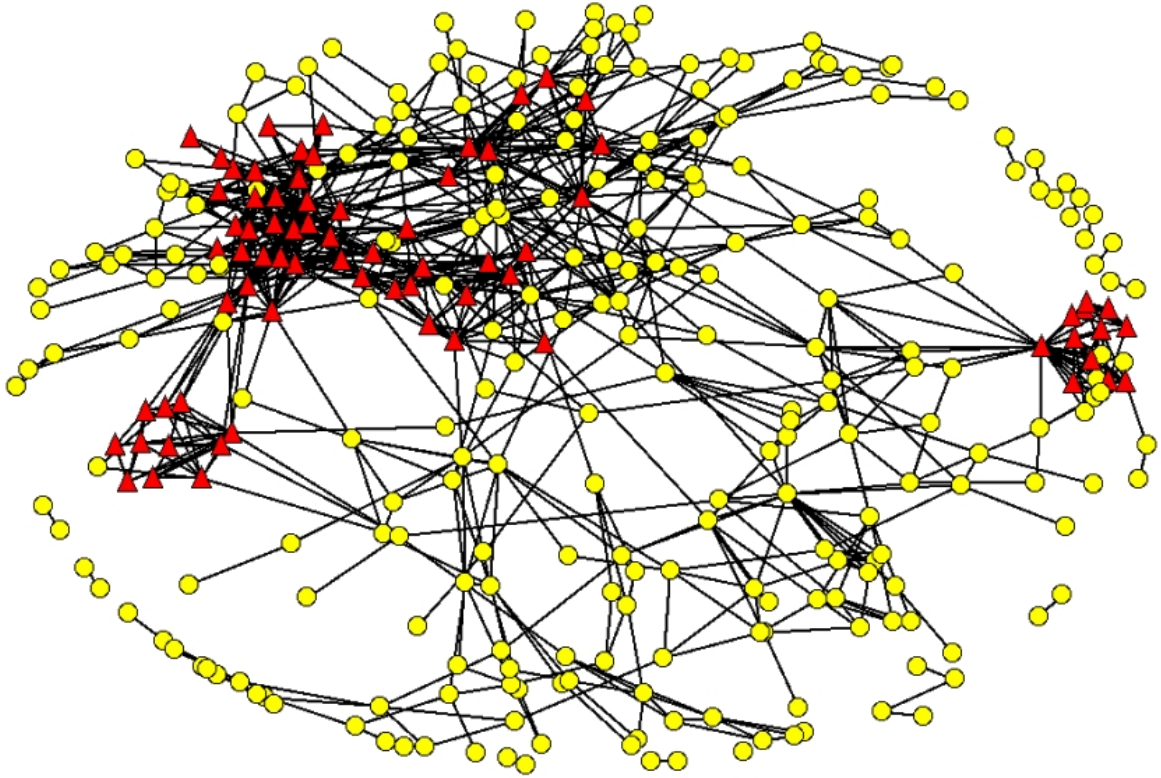


Figure 3.4: Graph induced by the attribute set $\{search, rank\}$ from the collaboration graph shown in Figure 1.2.

the structural correlation of S , $\epsilon(S)$, is given by:

$$\epsilon(S) = \frac{\kappa(S)}{|\mathcal{G}(S)|} \quad (3.2)$$

The structural correlations of the attribute sets $\{A\}$, $\{C\}$, and $\{A, B\}$ are 0.82, 0, and 1, respectively. Therefore, we say that the $\{A, B\}$ is more correlated with the formation of dense subgraphs than $\{A\}$, and there is no correlation between $\{C\}$ and dense subgraphs in the graph shown in Figure 1.1. Considering the graph of Figure 1.2, $\epsilon(\{search, rank\})$ is equal to 0.19.

We propose the concept of structural correlation to assess how vertex attributes induce dense subgraphs in an attributed graph. Therefore, it enables the identification of attribute sets that are significantly related to the formation of dense subgraphs. In Chapter 4 we describe algorithms for computing the structural correlation of attribute sets efficiently. Chapter 5 presents case studies on the application of the structural correlation to real datasets.

In the next section, we discuss how the structural correlation can be affected by the input graph, the attribute set support, and the dense subgraph parameters. Such relationships turn it difficult to determine what is a high or a low value of structural correlation in a given scenario. Moreover, the comparison of two attribute sets in terms of structural correlation may be biased if we do not take into account the support of such attribute sets. In order to address these issues, we propose two normalization approaches based on the expected structural correlation of attribute sets.

3.2 Normalized Structural Correlation

According to Definition 6, the structural correlation is the probability of a vertex with a given attribute set be covered by a dense subgraph. However, given the structural correlation of an attribute set, how can we evaluate it? In other words, what can be considered a high or low structural correlation? And how can we compare the structural correlation of different attribute sets from the same or even from distinct graphs? In this section, we address such questions by proposing null models for the structural correlation. Null models provide the expected properties of a dataset given a set of premises or assumptions. They have been applied in the identification and evaluation of graph patterns, such as motifs [Alon, 2006] and communities [Newman, 2004]. In the specific case of the structural correlation, a null model returns what is the expected structural correlation of an attribute set assuming that the correlation between vertex attributes and dense subgraphs is random. We call *normalized structural correlation* a function that measures how the structural correlation of an attribute set deviates from its expected value.

In a hypothetical scenario where there is no correlation between vertex attributes and dense subgraphs, one could expect that the structural correlation of any attribute set is 0. Nevertheless, the structural correlation does not depend only on how vertex attributes are organized. The graph topology, the attribute set support and the quasi-clique parameters can affect the structural correlation significantly and must be considered by null models for the structural correlation function.

The topology of the input graph has an important impact on the structural correlation of its attribute sets. We can illustrate the relationship between the graph topology and the structural correlation by comparing the graphs shown in Figures 1.1 and 3.5, which have the same number of vertices. In the graph from Figure 3.5, vertices are very well connected, therefore any attribute configuration in such graph will lead to high structural correlation values. In particular, if the minimum size of a dense

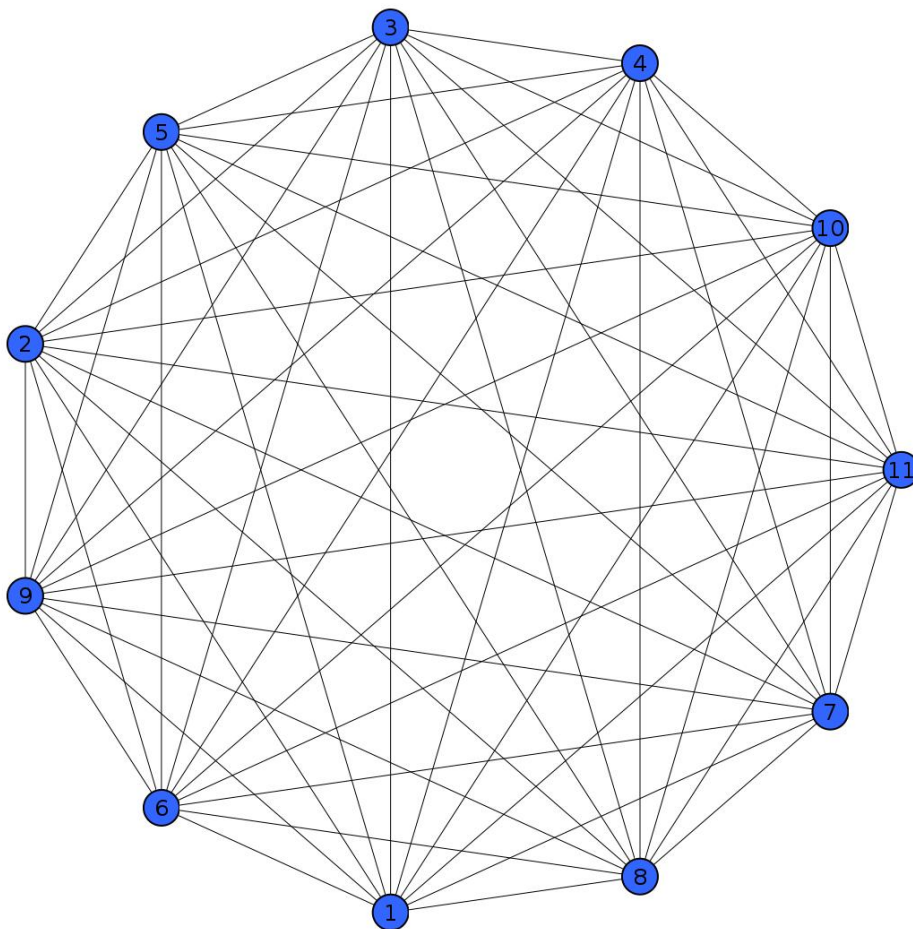


Figure 3.5: Clique with the same number of vertices that the graph shown in Figure 1.1

subgraph is 4 and we assign a hypothetical attribute to 4 randomly selected vertices from the graph shown in Figure 3.5, this attribute will have a structural correlation of 1. On the other hand, if we repeat the same experiment using the graph from Figure 1.1, there will be 330 possible settings with an average structural correlation of 0.05.

The expected structural correlation depends on the intrinsic dense subgraph organization from the input graph. In the case of the graph shown in Figure 3.5, for example, we can say that the structural correlation of its attribute sets is expected to be higher than in the graph from Figure 1.1.

The structural correlation is also affected by the support of the attribute sets. In general, the more frequent attribute sets are the more likely they present a high structural correlation. Considering the example graph shown in Figure 1.1, Table 3.2 presents the expected structural correlation (i.e., the average structural correlation) of a random subgraph of size σ , for σ varying from 4 to 11. We can notice that the

larger is σ , the higher is the expected structural correlation. Since the support of the attribute sets has a positive impact on the structural correlation, it is necessary to consider this impact in the design of null models for the structural correlation.

σ	# possible settings	ϵ_{exp}	sim- ϵ_{exp}	max- ϵ_{exp}
4	330	0.05	0.06	0.32
5	462	0.16	0.2	0.47
6	462	0.30	0.46	0.59
7	330	0.46	0.48	0.69
8	165	0.60	0.70	0.75
9	55	0.70	0.70	0.80
10	11	0.77	0.55	0.81
11	1	0.82	0.82	0.82

Table 3.2: Number of possible settings, expected structural correlation (ϵ_{exp}), simulation-based structural correlation (sim- ϵ_{exp}), and analytical normalized structural correlation (max- ϵ_{exp}) for N randomly selected vertices from the graph 1.1

Moreover, the dense subgraph parameters can also affect the structural correlation of attribute sets. In this work, we apply quasi-cliques as a definition for dense subgraphs (see Section 2.2). The parameters used in the identification of quasi-cliques are the minimum size (min_size) and density (γ_{min}). In general, the larger is the minimum size of quasi-cliques and the highest is the minimum density, the lower is the probability of finding a quasi-clique in the graph and, as a consequence, the lower will be the structural correlation. In the example graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1, if min_size and γ_{min} are set to 4 and 0.5, respectively, the structural correlation of the attribute A is 0.82. However, if we increase min_size to 5, the structural correlation of A is 0.54 and if we increase γ_{min} to 0.7, the structural correlation of A is reduced to 0.36. Since the dense subgraph parameters affect both the structural correlation of the attribute sets and their expected values, by comparing the structural correlation and the expected structural correlation, we may provide results that are less sensitive to specific dense subgraph parameters.

In order to compute the expected structural correlation of attribute sets, considering the impact of the input graph, the attribute set support, and the dense subgraph parameters, we propose two *null models*. Such models give an estimate of the expected structural correlation (ϵ_{exp}) of an attribute set with a support σ in a graph \mathcal{G} according to the quasi-clique parameters (γ_{min} and min_size).

We assume that the input graph \mathcal{G} comprises the object of interest, i.e., it is the “population” graph. Assume that we are given the attribute set support value $\sigma(S)$ (independent of the actual attribute set S). To compute the expected structural

correlation, our sample space is the set of all vertex subsets of size $\sigma(S)$ drawn randomly from \mathcal{G} . The statistic of interest is the mean structural correlation value, ϵ_{exp} . That is, the expected probability that a random vertex in a given sample induces dense subgraphs (quasi-cliques) in that sample of size $\sigma(S)$. The quasi-clique parameters, γ_{min} and min_size , are assumed to be fixed as well.

Since an analytical formulation for the expected structural correlation is not known, its exact computation requires averaging the structural correlation over the complete set of possible settings of a hypothetical attribute to random vertices from the input graph. The number of possible settings is $\binom{|V|}{\sigma}$, where σ is the support of the attribute set, a prohibitive computation for real large graphs. An alternative solution is to generate a limited number (r) of random settings and compute the average structural correlation as an approximation for the expected structural correlation. Algorithm 3 is a high-level description of such alternative, which we call *simulation null model for the structural correlation*. It applies a function *random-vertices* that selects σ random vertices from \mathcal{G} . Each selected vertex is checked to be in a quasi-clique, according to the quasi-clique parameters, through the *is-in-quasi-clique* function. The average fraction of vertices in quasi-cliques for r simulations is returned as the expected structural correlation. We define the simulation-based normalized structural correlation as follows.

Algorithm 3 Simulation Null Model for the Structural Correlation Algorithm

INPUT: $\mathcal{G}, \sigma, \gamma_{min}, min_size, r$
OUTPUT: $sim-\epsilon_{exp}$

- 1: $i \leftarrow 0$
- 2: $sim-\epsilon_{exp} \leftarrow 0$
- 3: **while** $i < r$ **do**
- 4: $V \leftarrow random-vertices(\mathcal{G}, \sigma)$
- 5: $n \leftarrow 0$
- 6: **for all** $v \in V$ **do**
- 7: **if** *is-in-quasi-clique*($v, V, \gamma_{min}, min_size$) **then**
- 8: $n \leftarrow n + 1$
- 9: **end if**
- 10: **end for**
- 11: $sim-\epsilon_{exp} \leftarrow \epsilon_{exp} + (n/\sigma)$
- 12: $i \leftarrow i + 1$
- 13: **end while**
- 14: $sim-\epsilon_{exp} \leftarrow sim-\epsilon_{exp}/r$

DEFINITION 7. (*Simulation-based normalized structural correlation*).
Given an attribute set S with support $\sigma(S)$, the simulation-based expected structural correlation of S is given by:

$$\delta_1(S) = \frac{\epsilon(S)}{sim-\epsilon_{exp}(\sigma(S))} \quad (3.3)$$

Although the simulation null model constitutes a much faster strategy than the complete enumeration of all possible vertex settings to compute the expected structural correlation, it may still be too expensive in real contexts. Therefore, we propose a second more efficient analytical null model for computing the expected structural correlation. The *analytical null model for the structural correlation* gives a theoretical upper bound on the structural correlation of an attribute set based on the quasi-clique definition. The idea is that a vertex must have a minimum degree of $\lceil \gamma_{min} \cdot (min_size - 1) \rceil$ in order to be member of a γ_{min} -quasi-clique of minimum size min_size . Consequently, the probability of a vertex to have a degree of $\lceil \gamma_{min} \cdot (min_size - 1) \rceil$ in a random subgraph of size σ from \mathcal{G} gives an upper bound on the expected structural correlation.

Given a randomly selected size σ subgraph \mathcal{G}_σ from \mathcal{G} , the degrees of v in \mathcal{G} and \mathcal{G}_σ are related according to Theorem 3.

THEOREM 3. (*Probability of a vertex that have a degree α in \mathcal{G} to have a degree β in \mathcal{G}_σ*). *If a random vertex v from \mathcal{G} with degree α is selected to be part of \mathcal{G}_σ , the probability of such vertex to have a degree β in \mathcal{G}_σ is given by the following binomial function:*

$$F(\alpha, \beta, \rho) = \binom{\alpha}{\beta} \cdot \rho^\beta \cdot (1 - \rho)^{\alpha - \beta} \quad (3.4)$$

where ρ is the probability of a specific vertex u from \mathcal{G} to be in \mathcal{G}_σ , if v is already taken:

$$\rho = \frac{\sigma - 1}{|\mathcal{V}| - 1} \quad (3.5)$$

Proof sketch. *There are α vertices adjacent to v in \mathcal{G} , thus, the probability of v to have a degree of exactly β in \mathcal{G}_σ is the probability of selecting β out of α vertices to be part of \mathcal{G}_σ . Since v is already selected, the probability of selecting any remaining vertex from \mathcal{G} is given by equation 3.5.*

Based on Theorem 3, we define an upper bound on the expected structural correlation, which is the probability of a vertex to have a degree of, at least, $\lceil \gamma_{min} \cdot (min_size - 1) \rceil$ in \mathcal{G}_σ .

THEOREM 4. (*Upper bound on the expected structural correlation*). *Given the quasi-clique parameters γ_{min} and min_size , the expected structural correlation of an attribute set with support σ is upper bounded by:*

$$max\text{-}\epsilon_{exp}(\sigma) = \sum_{\alpha=z}^m p(\alpha) \cdot \sum_{\beta=z}^{\alpha} F(\alpha, \beta, \rho) \quad (3.6)$$

where $z = \lceil \gamma_{min} \cdot (min_size - 1) \rceil$, m is the maximum degree of vertices from \mathcal{G} , and p is the degree distribution of \mathcal{G} .

Proof sketch. Given a vertex with degree α in \mathcal{G} , the probability of such vertex to have a degree of, at least, $\lceil \gamma_{min} \cdot (min_size - 1) \rceil$ in \mathcal{G}_σ is the sum of the expression 3.4 over the degree interval from $\lceil \gamma_{min} \cdot (min_size - 1) \rceil$ to α . If we multiply this sum by the probability of a vertex with degree α from \mathcal{G} to be in \mathcal{G}_σ , i.e., $p(\alpha)$, it gives the probability of any vertex with degree α from \mathcal{G} to have a degree of, at least, $\lceil \gamma_{min} \cdot (min_size - 1) \rceil$ in \mathcal{G}_σ . Equation 3.6 is the sum of such products over the vertex degrees higher than $\lceil \gamma_{min} \cdot (min_size - 1) \rceil$.

Similarly to the simulation model, we define the analytical normalized structural correlation as follows:

DEFINITION 8. (Analytical normalized structural correlation). Given an attribute set S with support $\sigma(S)$, the analytical structural correlation is given by:

$$\delta_2(S) = \frac{\epsilon(S)}{\max\text{-}\epsilon_{exp}(\sigma(S))} \quad (3.7)$$

Table 3.2 shows the simulation-based expected ϵ ($\text{sim-}\epsilon_{exp}$) and the upper bound on the expected structural correlation ($\text{max-}\epsilon_{exp}$) for the example graph shown in Figure 1.1 varying the value of σ from 4 to 11. The value of r was set to 10% of the number of possible settings. Considering the attributes presented in Table 1.1, $\delta_1(\{A\})$, $\delta_1(\{C\})$, and $\delta_1(\{A, B\})$ are equal to 1, 0, and 2.08, and $\delta_2(\{A\})$, $\delta_2(\{C\})$, and $\delta_2(\{A, B\})$ are equal to 1, 0, and 1.69, respectively. In the Section 5.1, we study the normalized structural correlation of attribute sets from real databases.

It is important to notice that the function $\text{max-}\epsilon_{exp}$ is monotonically non-decreasing, i.e. $\text{max-}\epsilon_{exp}(\sigma_1)$ is greater than $\text{max-}\epsilon_{exp}(\sigma_2)$ if and only if $\sigma_1 \geq \sigma_2$. We also assume that $\text{sim-}\epsilon_{exp}$ is monotonically non-decreasing for sufficiently high values of r . Such property will be applied further in this work.

3.3 Structural Correlation Patterns

The structural correlation is a property that correlates attribute sets and dense subgraphs in an attribute graph. Therefore, such property provides an analysis at the level of the attribute sets. Nevertheless, in several scenarios, it is relevant to provide also knowledge at the level of the dense subgraphs induced by attribute sets. We call

structural correlation pattern a dense subgraph that is homogeneous in terms of an attribute set.

DEFINITION 9. (*Structural correlation pattern*). *A structural correlation pattern is a pair (S, V) , where S is an attribute set ($S \subseteq \mathcal{A}$), and V is a quasi-clique from the graph induced by S ($V \subseteq \mathcal{V}(S)$), given the quasi-clique parameters γ_{min} and min_size .*

Considering the attributed graph of Figure 1.1, for which the vertex attributes are shown in Table 1.1, $(\{A, B\}, \{6, 7, 8, 9, 10, 11\})$ is an example of a structural correlation pattern. Figure 1.4 shows the dense subgraph composed by vertices in such pattern. The subgraph from Figure 1.5 represents an example of a real structural correlation pattern, with the keywords “search” and “web” as attributes, from the collaboration graph shown in Figure 1.2.

An important aspect related to the enumeration of structural correlation patterns is its computational cost, which may be prohibitive for large datasets. For a set of vertex attributes \mathcal{A} , the number of possible non-empty attribute sets is $2^{|\mathcal{A}|} - 1$. Moreover, the search space of non-empty quasi-cliques has, in the worst case, $2^{|\mathcal{V}|} - 1$ combinations of vertices, as described in Section 2.2. Therefore, the number of structural correlation patterns is $(2^{|\mathcal{A}|} - 1) \cdot (2^{|\mathcal{V}|} - 1)$, in the worst case. In order to limit both the search space and the number of structural correlation patterns returned to the user, we propose the identification of only the structural correlation patterns for which their respective attribute sets satisfy minimum support and structural correlation thresholds. Our argument is that in real graphs a great part of the structural correlation patterns do not constitute good representatives for the structural correlation at the attribute set level. In other words, most of the structural correlation patterns do not generalize a significant correlation between attribute sets and dense subgraphs in the input graph. In the next section, we formalize two versions of the structural correlation pattern mining problem, which aggregate both the concept of structural correlation and structural correlation pattern into new data mining tasks.

3.4 Structural Correlation Pattern Mining Problem

The *structural correlation pattern mining problem* is a new graph mining problem proposed in this work. It formalizes the general idea of extracting knowledge about how vertex attributes and dense subgraphs are correlated in attributed graphs. Therefore, in this section, we finish the important process of turning an abstract information need into a well-defined graph pattern.

In Section 3.1, we proposed the structural correlation as a measure of how a given attribute set induces dense subgraphs in an attributed graph and argued that such function provides knowledge at the level of attribute sets. Moreover, in Section 3.3, we defined structural correlation patterns as a materialization of the structural correlation at the level of dense subgraphs. At this point, we put the structural correlation and structural correlation pattern definitions together in order to provide knowledge at both levels.

In general terms, the structural correlation pattern mining comprises the identification of: (1) The attribute sets significantly correlated with the formation of dense subgraphs, and (2) the dense subgraphs induced by such attribute sets.

DEFINITION 10. (*Structural correlation pattern mining problem*). *Given an attributed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{F})$, a minimum support threshold σ_{min} , minimum quasi-clique density γ_{min} and size min_size , and a minimum structural correlation ϵ_{min} , the structural correlation pattern mining consists of identifying the set of structural correlation patterns (S, V) from \mathcal{G} , such that S is an attribute set for which $\sigma(S) \geq \sigma_{min}$, $\epsilon(S) \geq \epsilon_{min}$, and V is a γ_{min} -quasi-clique for which $V \subseteq \mathcal{V}(S)$ and $|V| \geq min_size$.*

Besides the input graph \mathcal{G} , the structural correlation pattern mining is based on 4 parameters which act as restrictions for the patterns to be generated. Therefore, it is relevant to describe the role played by each one of these parameters, which can be summarized in terms of three desired effects: ensuring the significance, scaling down the number of patterns, and reducing the computational cost of the enumeration of these patterns:

- σ_{min} : Regarding the significance of the structural correlation, the σ_{min} parameter prunes infrequent attribute sets, what is desired, since there might be not enough evidence of the structural correlation of them. Moreover, σ_{min} plays an important role on scaling down the number of attribute sets delivered to the user because a huge amount of patterns may be overwhelming in practical applications. But σ_{min} reduces not only the number of patterns generated, it also reduces the number of attribute sets for which the structural correlation will be computed, diminishing the computational cost of the structural correlation pattern mining task both in terms of processing time and memory requirements.
- γ_{min} and min_size : The minimum quasi-clique density and size avoid the generation of quasi-cliques which are not significant because they are too small and/or sparse. The quasi-clique parameters also shrink the number of patterns generated,

since dense and large quasi-cliques are expected to be infrequent in real graphs. Similar to the σ_{min} parameter, the reduction of the number of quasi-cliques also affects the computational cost of the quasi-clique discovery, leveraging the pruning capacity of the rules described in Section 2.2.

- ϵ_{min} : The minimum structural correlation is a new parameter that is specific for the structural correlation pattern mining problem. In terms of significance, by setting ϵ_{min} , the user may specify what is considered a representative structural correlation in the specific application scenario, distinguishing relevant patterns from the not relevant ones. The number of patterns delivered to the user can also be reduced according to ϵ_{min} , what enables the user to regulate the number of patterns generated according to its needs. Regarding the computational cost, in Section 4.3, we describe pruning techniques based on ϵ_{min} .

From a data mining perspective, the structural correlation pattern mining integrates two existing data mining problems: the frequent itemset mining (see Section 2.3) and the quasi-clique mining (see Section 2.2). Attribute sets can be seen as itemsets and are pruned based on a minimum support threshold. Structural correlation patterns are quasi-cliques induced by attribute sets. The concept of structural correlation works as an aggregating element, introducing novel knowledge regarding how vertex attributes induce dense subgraphs in an attributed graph. Moreover, the structural correlation pattern mining problem brings new algorithmic challenges in terms of efficiency and scalability requirements, which will be discussed in Chapter 4.

In order to exemplify the structural correlation pattern mining task, we consider the attributed graph shown in Figure 1.1, for which the vertex attributes are shown in Table 1.1. The parameters σ_{min} , γ_{min} , min_size and ϵ_{min} are set to 3, 0.6, 4, and 0.5, respectively. Table 3.3 shows the attribute sets discovered, with their respective supports (σ), structural coverages (κ), and structural correlations ϵ . In Table 3.4, we present the structural correlation patterns discovered, with their respective sizes (size) and densities (γ).

attribute set	σ	κ	ϵ
A	11	9	0.82
B	6	6	1.0
A, B	6	6	1.0

Table 3.3: Attribute sets, with their respective values of σ , κ , and ϵ , from the graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1 if $\sigma_{min} = 3$, $\gamma_{min} = 0.6$, $min_size = 4$, and $\epsilon_{min} = 0.5$

pattern	size	γ
$(\{A\}, \{6, 7, 8, 9, 10, 11\})$	6	0.60
$(\{A\}, \{3, 4, 5, 6\})$	4	1
$(\{A\}, \{3, 4, 6, 7\})$	4	0.67
$(\{A\}, \{3, 5, 6, 7\})$	4	0.67
$(\{A\}, \{3, 6, 7, 8\})$	4	0.67
$(\{B\}, \{6, 7, 8, 9, 10, 11\})$	6	0.60
$(\{A, B\}, \{6, 7, 8, 9, 10, 11\})$	6	0.60

Table 3.4: Structural correlation patterns, with their respective sizes (size) and densities γ , from the graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1 if $\sigma_{min} = 3$, $\gamma_{min} = 0.6$, $min_size = 4$, and $\epsilon_{min} = 0.5$

Similar to the quasi-clique mining and the frequent itemset mining, counting the number of structural correlation patterns is $\#P$ -hard and enumerating such patterns is an NP-hard problem, as stated by the following theorem.

THEOREM 5. (*Structural correlation pattern mining problem complexity*).

The problem of counting the structural correlation patterns from a graph \mathcal{G} is $\#P$ -hard and the problem of enumerating the structural correlation patterns from \mathcal{G} is NP-hard.

Proof sketch. *It can be proved by restriction. If we set a generic attribute a to each vertex from \mathcal{G} , set $\mathcal{F}(v) = \{a\}$ for every $v \in \mathcal{V}$, and set σ_{min} to 1 and ϵ_{min} to 0, then the structural correlation pattern mining problem becomes equivalent to the quasi-clique mining problem, which is proved to be $\#P$ -hard (see Theorem 1). It is known that NP-hard enumeration problems are associated to $\#P$ -hard counting problems [Yang, 2004].*

The application of the structural correlation pattern mining task to large databases requires efficient and scalable algorithms. In Chapter 4, we study algorithms for the structural correlation pattern mining problem.

In Section 3.2, we discussed how the structural correlation can be normalized through null models that give the expected structural correlation of a given attribute set. Following the same idea, we may also define a *normalized structural correlation pattern mining problem*.

DEFINITION 11. (*Normalized structural correlation pattern mining problem*).

Given an attributed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{F})$, a minimum support threshold σ_{min} , minimum quasi-clique density γ_{min} and size min_size , a minimum structural correlation ϵ_{min} and a minimum normalized structural correlation δ_{min} , the normalized structural correlation pattern mining consists of identifying the set of structural correlation patterns (S, V) from \mathcal{G} , such that S is an attribute set for which $\sigma(S) \geq \sigma_{min}$,

$\epsilon(S) \geq \epsilon_{min}$, $\delta(S) \geq \delta_{min}$, and V is a γ -quasi-clique for which $V \subseteq \mathcal{V}(S)$ and $|V| \geq min_size$.

The only difference between the standard and the normalized version of the structural correlation pattern mining problem is that the second one applies a minimum normalized structural correlation threshold (δ_{min}). The value of δ can be based on the simulation or the analytical null model (see Section 3.2). Table 3.5 shows the attribute sets discovered from the graph presented in Figure 1.1, with their respective supports (σ), structural coverages (κ), structural correlations ϵ , upper bounds on the expected structural correlations ($max\text{-}\epsilon_{exp}$), and analytical normalized structural correlations (δ_2). The parameters used are the same of the previous example. The minimum normalized structural correlation threshold δ_{min} is set to 1.0. The structural correlation patterns for each attribute set, with their respective sizes (size) and densities (γ), can be found in Table 3.6.

attribute set	σ	κ	ϵ	$max\text{-}\epsilon_{exp}$	δ_2
B	6	6	1.0	0.59	1.69
A, B	6	6	1.0	0.59	1.69

Table 3.5: Attribute sets, with their respective values of σ , κ , ϵ , $max\text{-}\epsilon_{exp}$, and δ_2 , from the graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1 if $\sigma_{min} = 3$, $\gamma_{min} = 0.6$, $min_size = 4$, and $\delta_{min} = 1.0$

pattern	size	γ
$(\{B\}, \{6, 7, 8, 9, 10, 11\})$	6	0.60
$(\{A, B\}, \{6, 7, 8, 9, 10, 11\})$	6	0.60

Table 3.6: Structural correlation patterns, with their respective sizes (size) and densities γ , from the graph shown in Figure 1.1, for which the vertex attributes are presented in Table 1.1 if $\sigma_{min} = 3$, $\gamma_{min} = 0.6$, $min_size = 4$, and $\delta_{min} = 1$

Counting the number of normalized structural correlation patterns and enumerating such patterns are also $\#$ -P-hard and NP-hard problems, respectively. Such proof is very similar to the one presented in Theorem 5 (setting δ_{min} to 0, instead of ϵ_{min}) and will be omitted.

This section formalized the structural correlation pattern mining problem, which is a new graph mining problem proposed by this work. Along this chapter, we discussed the problem of correlating vertex attributes and the formation of dense subgraphs starting from the attribute set level and then considering the dense subgraphs that express such a correlation. The concepts proposed were illustrated through both instructive

and real examples whenever possible. In the next chapter, we study the structural correlation pattern mining problem from an algorithmic perspective, designing efficient and scalable algorithms in order to enable the analysis of real large graphs.

Chapter 4

Structural Correlation Pattern Mining: Algorithms

This chapter presents algorithms for structural correlation pattern mining. In the previous chapter, we defined the structural correlation and the normalized structural correlation pattern mining problems. Moreover, we showed that such problems bring interesting computational challenges in terms of performance, since they belong to the class of NP-hard problems. In order to enable the processing of large graphs in a feasible time, we propose several techniques for efficient structural correlation pattern mining.

We start with a naive algorithm, described in Section 4.1, that is a straightforward combination of a frequent itemset mining and a quasi-clique mining algorithm. In Section 4.2 we focus on the problem of computing the structural correlation of an attribute set, which is a subproblem of both the structural correlation and the normalized structural correlation pattern mining problems. Two search strategies for computing the structural correlation are presented.

In Section 4.3, we propose pruning techniques for attribute sets in the structural correlation pattern mining. Such techniques have as objective to enhance the performance of structural correlation pattern mining algorithms without compromising their correctness. Section 4.4 proposes the computation of the structural correlation using sampling. The idea is to estimate the structural correlation of an attribute set based on a sample of vertices from its induced graph. Section 4.5 studies how the reduction of the number of structural correlation patterns to be identified may lead to a significant reduction of the time required to enumerate such patterns. We propose the identification of the top-k structural correlation patterns in terms of size and density, where k is a user-defined parameter.

Finally, in Section 4.7, we design parallel algorithms for structural correlation pattern mining. The popularization of multi-core computers in the recent years motivates the development of algorithms able to exploit the shared memory parallelism in a scalable fashion. The proposed algorithms are extensions of the sequential algorithms for structural correlation pattern mining to a multi-core environment and may enable the analysis of even larger graphs.

4.1 Naive Algorithm

The structural correlation pattern mining can be seen as a combination of two existing data mining problems: the quasi-clique and the frequent itemset mining. Therefore, a quasi-clique and a frequent itemset mining algorithm may be integrated into a naive algorithm for structural correlation pattern mining. Algorithm 4 is a high-level description of this naive algorithm.

The naive algorithm for structural correlation pattern mining receives the attributed graph \mathcal{G} , the minimum support threshold σ_{min} , the minimum density threshold γ_{min} , the minimum size of a dense subgraph min_size , the minimum structural correlation threshold ϵ_{min} , and the minimum normalized structural correlation threshold δ_{min} as parameters, and gives as output the set of structural correlation patterns \mathcal{P} from \mathcal{G} . It identifies the frequent attribute sets from the attributed graph using the function *frequent-attribute-sets*, which is a direct application of a frequent itemset mining algorithm. The specific frequent itemset mining algorithm applied in this work is the Eclat algorithm [Zaki, 2000]. For each frequent attribute set S , the algorithm identifies the set of quasi-cliques \mathcal{Q} from the induced graph $\mathcal{G}(S)$ using the function *quasi-cliques*, which is a direct application of a quasi-clique mining algorithm. We apply the Click algorithm [Liu and Wong, 2008] for quasi-clique mining. If the structural correlation of an attribute set S satisfies the minimum structural correlation and normalized structural correlation thresholds, a structural correlation pattern (S, q) , where q is a quasi-clique from $\mathcal{G}(S)$, is included into the set of structural correlation patterns (\mathcal{P}) for each quasi-clique $q \in \mathcal{Q}$. Since the naive algorithm receives both ϵ_{min} and δ_{min} , it solves both the standard and the normalized version of the structural correlation pattern mining problem.

The proposed algorithms are straightforward extensions of existing data mining algorithms. Nevertheless, along the remaining of this chapter, we design new algorithms for the problems defined in the last chapter. Such algorithms are based on idea of mining structural correlation patterns in two steps. The first step computes the structural

Algorithm 4 Naive Algorithm For Structural Correlation Pattern Mining

INPUT: $\mathcal{G}, \sigma_{min}, \gamma_{min}, min_size, \epsilon_{min}$
OUTPUT: \mathcal{P}
1: $\mathcal{I} \leftarrow frequent_attribute_sets(\mathcal{G}, \sigma_{min})$
2: $\mathcal{P} \leftarrow \emptyset$
3: **for all** $S \in \mathcal{I}$ **do**
4: $\mathcal{Q} \leftarrow quasi_cliques(\mathcal{G}(S), \gamma_{min}, min_size)$
5: **if** $\epsilon(\mathcal{Q}, S) \geq \epsilon_{min}$ **then**
6: **for all** $q \in \mathcal{Q}$ **do**
7: $\mathcal{P} \leftarrow \mathcal{P} \cup (S, q)$
8: **end for**
9: **end if**
10: **end for**

correlation of the attribute sets. In the second step, the structural correlation patterns are identified. The first step can be performed more efficiently than the second one and we consider this a property in order to speedup the proposed algorithms. Moreover, in several scenarios, the enumeration of the complete set of structural correlation patterns may not be a requirement. In such scenarios, computing only the structural correlation of the attribute sets may result in significative performance gains in comparison to solving the original structural correlation pattern mining problem. Along this chapter, we also study the problem of identifying a smaller set of most interesting structural correlation patterns. In the next section, we discuss the computation of the structural correlation of attribute sets.

4.2 Computing the Structural Correlation

Computing the structural correlation is the first step in solving the structural correlation pattern mining and normalized structural correlation pattern mining problems (see Section 3.4). As discussed along chapter 3, the structural correlation is an analysis at the attribute set level, since it evaluates how a particular attribute set induces dense subgraphs in an attributed graph. The normalized structural correlation is a similar task, but it normalizes the structural correlation based on the expected structural correlation of a given attribute set (see Section 3.2).

The structural correlation of an attribute set S is the probability of a vertex to be in a dense subgraph in the graph induced by S ($\mathcal{G}(S)$). The naive algorithm, described in the last section, compute the structural correlation based on the complete set of dense subgraphs in $\mathcal{G}(S)$. However, the computation of the structural correlation of an attribute set does not require the enumeration of the complete set of dense subgraphs from its induced graph. For each vertex, the only necessary information is whether it is or not covered by, at least, one dense subgraph. Since a vertex can be member of

several dense subgraphs, we may compute the structural correlation of an attribute set more efficiently by not enumerating the complete set of dense subgraphs,

As discussed in Section 2.2, the search space of quasi-cliques can be represented by a set enumeration tree, such as the one shown in Figure 2.1. As long as enumerating the complete set of dense subgraphs is not necessarily a requirement, we should study different strategies for traversing the search space for dense subgraphs in order to check whether each vertex is, at least, in one dense subgraph. If a vertex set contains only vertices already known to be in dense subgraphs, it can be pruned. Since the set enumeration tree is a graph, traditional graph search strategies can be applied to examine which vertices are covered by dense subgraphs. A breadth-first search (BFS) algorithm for the computation of the structural correlation of an attribute set traverses the set enumeration tree in a breadth-first order, starting from the root and visiting the smaller vertex sets before the larger ones. Figure 4.1 shows how vertices from the enumeration tree presented in Figure 2.1 are visited in a breadth-first search. On the other hand, a depth-first search (DFS) strategy for computing the structural correlation extends vertex sets as much as possible. In Figure 4.2, we show how the set enumeration tree from Figure 2.1 is traversed in DFS.



Figure 4.1: Order of visit of candidate patterns in a BFS search

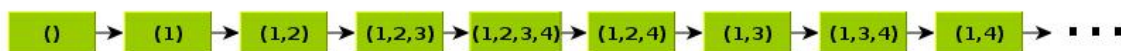


Figure 4.2: Order of visit of candidate patterns in a DFS search

Algorithm 5 describes, in high-level, the computation of the structural correlation function. Such algorithm receives an induced graph ($\mathcal{G}(S)$), a minimum density threshold (γ_{min}), the minimum size of dense subgraphs (min_size), and the minimum structural coverage threshold (κ_{min}) as parameters and returns the structural correlation (ϵ). The minimum structural coverage is the minimum number of vertices covered by dense subgraphs in the graph induced by a given attribute set and can be directly derived from ϵ_{min} . The function *vertex-pruning* integrates the vertex pruning rules described in Section 2.2 and returns those vertices that were not pruned. The set

Algorithm 5 structural-correlation**INPUT:** \mathcal{G} , γ_{min} , min_size , κ_{min} **OUTPUT:** ϵ

```

1:  $candExts(X) \leftarrow vertex\_pruning(\mathcal{V}(S), \gamma_{min}, min\_size)$ 
2: if  $|candExts(X)| \leq \kappa_{min}$  then
3:    $\epsilon \leftarrow 0$ 
4: else
5:   if  $searchStrategy = BFS$  then
6:      $K \leftarrow coverage\_BFS(candExts(X), \mathcal{G}(S), \gamma_{min}, min\_size)$ 
7:   else
8:     if  $searchStrategy = DFS$  then
9:        $K \leftarrow coverage\_DFS(candExts(X), \mathcal{G}(S), \gamma_{min}, min\_size)$ 
10:    end if
11:  end if
12:   $\epsilon \leftarrow |K|/|\mathcal{G}(S)|$ 
13: end if

```

$candExts(X)$ is initialized as the set of vertices $\mathcal{V}(S)$, excluding those ones pruned by the *vertex-pruning* function. The BFS or the DFS strategy can be applied according to the variable *searchStrategy*. The value of ϵ is the ratio of the size of the coverage (K) to the size of the induced subgraph ($\mathcal{G}(S)$). We examine in more detail the functions *coverage-BFS* and *coverage-DFS* in the remaining of this section.

Algorithm 6 coverage-BFS**INPUT:** $candExts(X)$, $\mathcal{G}(S)$, γ_{min} , min_size **OUTPUT:** K

```

1:  $X \leftarrow \emptyset$ 
2:  $qcCands \leftarrow \emptyset$ 
3:  $qcCands.enqueue((X, candExts(X)))$ 
4:  $K \leftarrow \emptyset$ 
5: while  $qcCands \neq \emptyset$  do
6:    $q \leftarrow qcCands.dequeue()$ 
7:   if  $candidate\_quasi\_clique\_pruning(q.X, q.candExts(X), \mathcal{G}(S), \gamma_{min}, min\_size) = FALSE$  then
8:     if  $is\_quasi\_clique(q.X \cup q.candExts(X), \gamma_{min}, min\_size)$  then
9:        $K \leftarrow K \cup (q.X \cup q.candExts(X))$ 
10:    else
11:      if  $is\_quasi\_clique(q.X, \gamma_{min}, min\_size)$  then
12:         $K \leftarrow K \cup q.X$ 
13:      end if
14:      for all  $v \in q.candExts(X)$  do
15:         $t.candExts(X) \leftarrow \{u \in q.candExts(X) | u > v\}$ 
16:         $t.X \leftarrow q.X \cup v$ 
17:        if  $t.X \cup t.candExts(X) \not\subseteq K$  then
18:           $qcCands.enqueue(t)$ 
19:        end if
20:      end for
21:    end if
22:  end if
23: end while

```

The function *coverage-BFS*, described by Algorithm 6, computes the coverage K using BFS. It receives the set $candExts(X)$, an induced graph $\mathcal{G}(S)$, and the quasi-clique parameters γ_{min} and min_size . The coverage K , composed by the vertices in $\mathcal{G}(S)$ that are covered by quasi-cliques, is returned as output. The data structure $qcCands$ is a queue initialized with the pair $(X, candExts(X))$. The set X is initialized

as empty. New patterns are inserted and processed in a first-in-first-out (FIFO) order. For each pair $q = (X, candExts)$ dequeued, the function *candidate-quasi-clique-pruning* applies the candidate quasi-clique pruning rules described in Section 2.2. This function returns TRUE if a vertex from the set $q.X$ is pruned and FALSE, otherwise. In case a vertex from $q.X$ is pruned, the pair q is pruned too, otherwise, a lookahead checking is performed. If $q.X \cup q.candExts(X)$ is a quasi-clique, then each vertex $v \in q.X \cup q.candExts(X)$ is included into the coverage set K . If X is a quasi-clique, then each vertex $v \in q.X$ is added to K . Moreover, new extensions of q are generated and those extensions that contain, at least, one vertex not in K are enqueued into *qcCands*. Such procedure is repeated until *qcCands* becomes empty.

Algorithm 7 coverage-DFS

INPUT: $candExts(X), \mathcal{G}(S), \gamma_{min}, min_size$
OUTPUT: K

```

1:  $X \leftarrow \emptyset$ 
2:  $qcCands \leftarrow \emptyset$ 
3:  $qcCands.push((X, candExts(X)))$ 
4:  $K \leftarrow \emptyset$ 
5: while  $qcCands \neq \emptyset$  do
6:    $q \leftarrow qcCands.pop()$ 
7:   if  $candidate-quasi-clique-pruning(q.X, q.candExts(X), \mathcal{G}(S), \gamma_{min}, min\_size) = FALSE$  then
8:     if  $is-quasi-clique(q.X \cup q.candExts(X), \gamma_{min}, min\_size)$  then
9:        $K \leftarrow K \cup (q.X \cup q.candExts(X))$ 
10:    else
11:      if  $is-quasi-clique(q.X, \gamma_{min}, min\_size)$  then
12:         $K \leftarrow K \cup q.X$ 
13:      end if
14:      for all  $v \in q.candExts(X)$  do
15:         $t.candExts(X) \leftarrow \{u \in q.candExts(X) | u > v\}$ 
16:         $t.X \leftarrow q.X \cup v$ 
17:        if  $t.X \cup t.candExts(X) \not\subseteq K$  then
18:           $qcCands.push(t)$ 
19:        end if
20:      end for
21:    end if
22:  end if
23: end while

```

The function *coverage-DFS* (Algorithm 7) is very similar to the function *coverage-BFS*, but while the first identifies the coverage K from an induced graph $\mathcal{G}(S)$ using DFS, the second one applies BFS. The data structure *qcCands*, used to manipulate the pairs $(X, candExts(X))$ is a stack, instead of a queue. Therefore, new pairs are pushed into and popped out of *qcCands* until it becomes empty.

The next sections present pruning, sampling, and parallelization strategies for structural correlation pattern mining. Such strategies may be combined with the search strategies presented in this section in the design of efficient algorithms. In Chapter 5 we study the performance of the algorithms described along this chapter using real datasets.

4.3 Pruning Techniques

In the last section, we presented an algorithm for computing the structural correlation of an attribute set. In this section, we propose pruning techniques for structural correlation pattern mining. The objective of these pruning techniques is to reduce the execution time of the structural correlation pattern mining algorithms without compromising its correctness.

A traditional approach for the enumeration of patterns that are combinations of smaller patterns (e.g., itemsets, sequences, subgraphs) is the use of a level-wise enumeration strategy. In the case of attribute sets, for example, an attribute set $\{A, B\}$ can be generated through the combination of the attribute sets $\{A\}$ and $\{B\}$. Figure 2.3 shows subset/superset relationships in a lattice representation of the search space of itemsets. Several algorithms that exploit such type of search space apply pruning techniques based on the anti-monotonicity property (see Definition 3). Nevertheless, the anti-monotonicity property does not hold for attribute sets in structural correlation pattern mining.

PROPOSITION 3. (*Anti-monotonicity or downward-closure property of attribute sets in the structural correlation pattern mining*). *The anti-monotonicity (or downward-closure) property does not hold for attribute sets in structural correlation pattern mining.*

Proof sketch. *We prove it by counterexample. Lets consider the example graph shown in Figure 1.1 and the parameters σ_{min} , γ_{min} , min_size , and ϵ_{min} be set to 3, 0.6, 4, and 0.9, respectively. In such setting, $\epsilon(\{A\}) = 0.82$ and $\epsilon(\{A, B\}) = 1$. Therefore, $(\{A\}, V)$ is not a structural correlation pattern for any $V \subseteq \mathcal{V}(A)$, but $(\{A, B\}, 6, 7, 8, 9, 10, 11)$ is a structural correlation pattern.*

Similar to the structural correlation pattern mining, the anti-monotonicity property also does not hold for attribute sets in normalized structural correlation pattern mining problem.

PROPOSITION 4. (*Anti-monotonicity or downward-closure property of attribute sets in the normalized structural correlation pattern mining*). *The anti-monotonicity (or downward-closure) property does not hold for attribute sets in structural correlation pattern mining.*

Proof sketch. *We prove it by counterexample. Lets consider the example graph shown in Figure 1.1 and the parameters σ_{min} , γ_{min} , min_size , and δ_{min} be set to 3, 0.6, 4, and 1.2, respectively. In such setting, $\delta_2(\{A\}) = 1$ and $\delta_2(\{A, B\}) = 1.64$. Therefore,*

$(\{A\}, V)$ is not a normalized structural correlation pattern for any $V \subseteq \mathcal{V}(A)$, but $(\{A, B\}, 6, 7, 8, 9, 10, 11)$ is a normalized structural correlation pattern.

As discussed in Section 2.2, several pruning strategies in data mining are based on the anti-monotonicity property. However, since the anti-monotonicity property does not hold for attribute sets in the standard and normalized versions of the structural correlation pattern mining, we define new specific pruning rules for structural correlation pattern mining. The first strategy allows the pruning of vertices during the level-wise enumeration of attribute sets.

THEOREM 6. (Vertex pruning for attribute sets). *Let K_{S_i} be the coverage of an attribute set S_i and K_{S_j} be the structural coverage of an attribute set S_j , if $S_i \subseteq S_j$, then $K_{S_j} \subseteq K_{S_i}$.*

Proof sketch. *It can be proved by contradiction. Lets suppose that there exists a vertex v such that $v \in K_{S_j}$ and $v \notin K_{S_i}$. Since $v \in K_{S_j}$, there exists a dense subgraph $V \subseteq \mathcal{V}(S_j)$, such that $v \in V$. Moreover, if $v \notin K_{S_i}$, there does not exist a dense subgraph $U \subseteq \mathcal{V}(S_i)$ such that $v \in U$. Nevertheless, if $S_i \subseteq S_j$, then $\mathcal{V}(S_j) \subseteq \mathcal{V}(S_i)$, what implies that $V \subseteq \mathcal{V}(S_i)$ (contradiction).*

Based on Theorem 6, we can prune any vertex that is not part of a dense subgraph in the graph induced by an attribute set of size i before extending it to generate attribute sets of size $i + 1$. Attribute sets can also be pruned based on an upper bound of the structural correlation function, as stated by Theorem 7.

THEOREM 7. (Attribute set pruning based on the upper bound of the structural correlation). *For two attribute sets S_i and S_j , if $S_i \subseteq S_j$ and $\sigma(S_j) \geq \sigma_{min}$, then $\epsilon(S_j) \leq \epsilon(S_i) \cdot |\mathcal{V}(S_i)| / \sigma_{min}$*

Proof sketch. *According to Theorem 6, $\epsilon(S_i) \cdot |\mathcal{V}(S_i)| \geq \epsilon(S_j) \cdot |\mathcal{V}(S_j)|$, since every vertex covered by a dense subgraph in $\mathcal{V}(S_j)$ is also covered by a dense subgraph in $\mathcal{V}(S_i)$. Moreover, since $\sigma(S_j) \geq \sigma_{min}$, $\epsilon(S_j)$ is upper bounded by $\epsilon(S_i) \cdot |\mathcal{V}(S_i)| / \sigma_{min}$ based on the definition of the structural correlation function ϵ (see Definition 6).*

We apply Theorem 7 like an apriori-based pruning [Agrawal and Srikant, 1994; Agrawal et al., 1993]. Given an attribute set S_i , of size i , if $\epsilon(S_i) \cdot |\mathcal{V}(S_i)| / \sigma_{min} < \epsilon_{min}$, then S_i is not included in the set of attribute sets to be combined for the generation of size $i + 1$ attribute sets. Theorem 7 guarantees that there does not exist an attribute set S_j , such that $S_i \subseteq S_j$ and $\epsilon(S_j) \geq \epsilon_{min}$. A similar pruning rule can be formulated based on the normalized structural correlation function definition.

THEOREM 8. (Attribute set pruning based on the upper bound of the normalized structural correlation). For two attribute sets S_i and S_j , if $S_i \subseteq S_j$, ϵ_{exp} is a monotonically non-decreasing, and $\sigma(S_j) \geq \sigma_{min}$, then $\delta(S_j) \leq \epsilon(S_i) \cdot |\mathcal{V}(S_i)| / (\epsilon_{exp}(\sigma_{min}) \cdot \sigma_{min})$

Proof sketch. According to Theorem 7, $\epsilon(S_j) \leq \epsilon(S_i) \cdot |\mathcal{V}(S_i)| / \sigma_{min}$. Furthermore, since $\sigma(S_j) \geq \sigma_{min}$ and ϵ_{exp} is monotonically non-decreasing, $\epsilon_{exp}(\sigma(S_j)) \geq \epsilon_{exp}(\sigma_{min})$. Therefore, $\delta(S_j) \leq \epsilon(S_i) \cdot |\mathcal{V}(S_i)| / (\epsilon_{exp}(\sigma_{min}) \cdot \sigma_{min})$.

Theorem 8 is applied similarly to Theorem 7. If $\delta(S_i) \cdot |\mathcal{V}(S_i)| / (\epsilon_{exp}(\sigma_{min}) \cdot \sigma_{min}) < \delta_{min}$, then the attribute set S_i , of size i , is not included in the set of attribute sets to be combined for the generation of size $i + 1$ attribute sets.

The proposed pruning techniques are applied in order to allow an efficient mining of structural correlation patterns. Pruning is the first effort in order to compute the structural correlation and identify structural correlation patterns efficiently. In the next section, we describe how the structural correlation may be computed using sampling.

4.4 Sampling

Sampling is a statistical technique for drawing conclusions about a population of interest based on a sample of such data [Wonnacott and Wonnacott, 1985]. In general, sampling is applied when the entire population is not known or is too large to be considered. A sample is a subset of the original population and it is expected to be informative about the total population.

In section 3.4, we described the structural correlation pattern mining and proved its NP-hardness. In particular, computing the structural correlation of attribute sets is a computationally expensive task. In this section, we study how sampling can be applied in order to reduce the execution time for computing the structural correlation. Since sampling techniques are subject to errors, it is important to provide guarantees regarding how close the estimation given is from the structural correlation based on the entire population.

There are several sampling techniques in the literature [Wonnacott and Wonnacott, 1985; Anderson and Finn, 1996]. Such techniques have as main objective to select samples that are representative of the population. We apply a traditional sampling strategy called *random sampling*, which is the random selection of n items from a population of size N . In a random sampling, every subset of size n has the same probability of being selected among the $\binom{N}{n}$ possible subsets. Random sampling is known to work very well in several scenarios, including frequent pattern mining [Zhao et al., 2006;

Zaki et al., 1997]. Since the structural correlation pattern mining applies a minimum support threshold in order to evaluate only the frequent attribute sets, we expect that sampling might work well in the structural correlation mining.

For a given attribute set S , the set of vertices from the graph induced by S ($\mathcal{V}(S)$) is defined as the population. Algorithm 5 computes the exact structural correlation of a given attribute set S using the entire population of vertices $\mathcal{V}(S)$. Nevertheless, we propose a faster computation of the structural correlation of an attribute set S using a random sample $Z \subseteq \mathcal{V}(S)$. Since the structural correlation function is a proportion, we estimate the structural correlation of S by the proportion of vertices from Z that are in dense subgraphs. The *margin of error*, which is a measure of the sampling error, of such a random sampling with a $100(1-\alpha)$ confidence interval is given by:

$$error = z_{\alpha/2} \sqrt{\frac{\epsilon(1-\epsilon)(|\mathcal{V}(S)| - |Z|)}{|Z|(|\mathcal{V}(S)| - 1)}} \quad (4.1)$$

where the value of $z_{\alpha/2}$ is the $1 - \alpha/2$ -quantile of a normal distribution.

Algorithm 8 is a high-level description of an algorithm for computing the structural correlation of an attribute set S using sampling. The algorithm receives the attribute set S , the attributed graph \mathcal{G} , the dense subgraph parameters γ_{min} and min_size , and the maximum margin of error accepted θ_{max} . The output of the algorithm is the structural correlation of S w.r.t. the input parameters. New random vertices are checked to be in dense subgraphs through the function *check-vertex-in-quasi-clique* until the margin of error θ_{max} is reached. Vertices known to be part of dense subgraphs are not checked more than one time. We decided not to show such optimization to keep the pseudo-code simple and concise.

The function *check-vertex-in-quasi-clique*, described by Algorithm 9, receives a vertex v , an induced graph $\mathcal{G}(S)$, and the dense subgraph parameters min_size and γ_{min} . It returns TRUE if v is in a quasi-clique in the induced graph $\mathcal{G}(S)$, and FALSE, otherwise. The set X is initialized with v and the set of candidate extensions of X , $candExts(X)$, is initialized with the vertices from $\mathcal{G}(S)$ ($\mathcal{V}(S)$). A vertex may be checked to be in a quasi-clique using a BFS or a DFS strategy. The function *find-quasi-clique-BFS* receives X , $candExts(X)$, $\mathcal{G}(S)$, γ_{min} , and min_size as parameters and returns TRUE if it finds, at least, one quasi-clique that is an extension of X by vertices from $candExts(X)$ and FALSE, otherwise. Similar to the function *coverage-BFS*, the function *find-quasi-clique-BFS*, described by Algorithm 10 applies a BFS strategy in the search for quasi-cliques. The function *find-quasi-clique-DFS*, which is described by Algorithm 11 applies a DFS strategy to find a quasi-clique containing v .

Algorithm 8 structural-correlation-with-sampling

INPUT: $S, \mathcal{G}, \gamma_{min}, min_size, \theta_{max}, \kappa_{min}$
OUTPUT: ϵ

- 1: $U \leftarrow vertex_pruning(\mathcal{V}(S), \gamma_{min}, min_size)$
- 2: **if** $|U| \leq \kappa_{min}$ **then**
- 3: $\epsilon \leftarrow -1$
- 4: **else**
- 5: $\theta \leftarrow 1$
- 6: $n \leftarrow 0$
- 7: $Z \leftarrow \emptyset$
- 8: $\kappa \leftarrow 0$
- 9: **while** $\theta > \theta_{max}$ **do**
- 10: $v \leftarrow new_random_vertex(\mathcal{G}(S))$
- 11: $Z \leftarrow Z \cup v$
- 12: **if** *check-vertex-in-quasi-clique*($v, \mathcal{G}(S), \gamma_{min}, min_size$) **then**
- 13: $\kappa \leftarrow \kappa + 1$
- 14: **end if**
- 15: $\theta \leftarrow margin_of_error(Z, \kappa, |\mathcal{G}(S)|)$
- 16: **end while**
- 17: $\epsilon \leftarrow \kappa/n$
- 18: **end if**

Algorithm 9 check-vertex-in-quasi-clique

INPUT: $v, \mathcal{G}(S), \gamma_{min}, min_size$
OUTPUT: is-in-qc

- 1: $candExts(X) \leftarrow \{u \in \mathcal{V}(S) | u \neq v\}$
- 2: $X \leftarrow \{v\}$
- 3: is-in-qc \leftarrow FALSE
- 4: **if** searchStrategy = BFS **then**
- 5: is-in-qc $\leftarrow find_quasi_clique_BFS(X, candExts(X), \mathcal{G}(S), \gamma_{min}, min_size)$
- 6: **else**
- 7: **if** searchStrategy = DFS **then**
- 8: is-in-qc $\leftarrow find_quasi_clique_DFS(X, candExts(X), \mathcal{G}(S), \gamma_{min}, min_size)$
- 9: **end if**
- 10: **end if**

In Section 5.3.2, we evaluate the sampling algorithm for computing the structural correlation proposed in this section in terms of both execution time and error. The next section discusses the problem of discovering structural correlation patterns and presents an algorithm for the discovery of the top structural correlation patterns from an attributed graph in terms of size and density.

4.5 Top-k Structural Correlation Patterns

A structural correlation pattern is a quasi-clique induced by a given attribute set, as described in Section 3.3. Structural correlation patterns are useful as representatives of the structural correlation at the level of dense subgraphs. Nevertheless, enumerating structural correlation patterns is a computationally expensive task, specially for large graphs. In this section, we study how to reduce the computational cost of enumerating the set of structural correlation patterns by restricting the output set to only the most interesting patterns.

Algorithm 10 find-quasi-clique-BFS

INPUT: $(X, \text{candExts}(X), \mathcal{G}(S), \gamma_{\min}, \text{min_size})$ **OUTPUT:** found

```

1:  $qcCands \leftarrow \emptyset$ 
2:  $qcCands.enqueue((X, \text{candExts}(X)))$ 
3: found  $\leftarrow$  FALSE
4: while  $qcCands \neq \emptyset$  do
5:    $q \leftarrow qcCands.dequeue()$ 
6:   if  $\text{candidate-quasi-clique-pruning}(q.X, q.\text{candExts}(X), \mathcal{G}(S), \gamma_{\min}, \text{min\_size}) = \text{FALSE}$  then
7:     if  $\text{is-quasi-clique}(q.X, \gamma_{\min}, \text{min\_size})$  then
8:       found  $\leftarrow$  TRUE
9:       return
10:    end if
11:    for all  $v \in q.\text{candExts}(X)$  do
12:       $t.\text{candExts}(X) \leftarrow \{u \in q.\text{candExts}(X) | u > v\}$ 
13:       $t.X \leftarrow q.X \cup v$ 
14:       $qcCands.enqueue(t)$ 
15:    end for
16:  end if
17: end while

```

Algorithm 11 find-quasi-clique-DFS

INPUT: $X, \text{candExts}(X), \mathcal{G}(S), \gamma_{\min}, \text{min_size})$ **OUTPUT:** found

```

1:  $qcCands \leftarrow \emptyset$ 
2:  $qcCands.push((X, \text{candExts}(X)))$ 
3:  $V \leftarrow \emptyset$ 
4: found  $\leftarrow$  FALSE
5: while  $qcCands \neq \emptyset$  do
6:    $q \leftarrow qcCands.pop()$ 
7:   if  $\text{candidate-quasi-clique-pruning}(q.X, q.\text{candExts}(X), \mathcal{G}(S), \gamma_{\min}, \text{min\_size}) = \text{FALSE}$  then
8:     if  $\text{is-quasi-clique}(q.X, \gamma_{\min}, \text{min\_size})$  then
9:       found  $\leftarrow$  TRUE
10:      return
11:    end if
12:    for all  $v \in q.\text{candExts}(X)$  do
13:       $t.\text{candExts}(X) \leftarrow \{u \in q.\text{candExts}(X) | u > v\}$ 
14:       $t.X \leftarrow q.X \cup v$ 
15:      if  $t.X \cup t.\text{candExts}(X) \not\subseteq V$  then
16:         $qcCands.push(t)$ 
17:      end if
18:    end for
19:  end if
20: end while

```

In Sections 4.2, 4.3, and 4.4, we focused on the problem of computing the structural correlation of attribute sets efficiently. Computing the structural correlation of attribute sets is the first step in structural correlation pattern mining. In particular, the proposed algorithms are based on the principle that it is not necessary to discover the complete set of dense subgraphs in order to assess the structural correlation of an attribute set S . However, if an attribute set S satisfies the minimum structural correlation threshold, according to the structural correlation pattern mining definition (see Section 3.4), the set of structural correlation patterns induced by S is generated as output.

We propose to restrict the number of structural correlation patterns returned to

Algorithm 12 top-k-structural-correlation-patterns

```

INPUT:  $\mathcal{G}(S)$ ,  $k$ ,  $\gamma_{min}$ ,  $min\_size$ 
OUTPUT:  $\mathcal{C}$ 
1:  $\mathcal{C} \leftarrow \emptyset$ 
2:  $qcCands \leftarrow \emptyset$ 
3:  $q.X \leftarrow \emptyset$ 
4:  $q.candExts(X) \leftarrow vertex\_pruning(\mathcal{V}(S), \gamma_{min}, min\_size)$ 
5:  $qcCands.push(q)$ 
6: while  $qcCands \neq \emptyset$  do
7:    $q \leftarrow qcCands.pop()$ 
8:   if  $|q.X| + |q.candExts(X)| \geq min\_size$  then
9:     if  $candidate\_quasi\_clique\_pruning(q.X, q.candExts(X), \mathcal{G}(S), \gamma_{min}, min\_size) = \text{FALSE}$  then
10:      if  $|q.X| + |q.candExts(X)| \geq min\_size$  then
11:        if  $is\_quasi\_clique(q.X \cup q.candExts(X), \gamma_{min}, min\_size)$  then
12:           $min\_size \leftarrow try\_to\_update\_top\_patterns(q.X \cup q.candExts(X), \mathcal{C}, min\_size)$ 
13:        else
14:          if  $is\_quasi\_clique(q.X, \gamma_{min}, min\_size)$  then
15:             $min\_size \leftarrow try\_to\_update\_top\_patterns(q.X, \mathcal{C}, min\_size)$ 
16:          end if
17:          for all  $v \in q.candExts(X)$  do
18:             $t.candExts(X) \leftarrow \{u \in q.candExts(X) | u > v\}$ 
19:             $t.X \leftarrow q.X \cup v$ 
20:             $qcCands.push(t)$ 
21:          end for
22:        end if
23:      end if
24:    end if
25:  end if
26: end while

```

the user for two reasons:

- **Performance:** The enumeration of the structural correlation patterns induced by a given attribute set S is the enumeration of the complete set of quasi-cliques from the graph induced by S . For frequent attribute sets, the induced graphs may be very large and enumerating their quasi-cliques may be prohibitive.
- **Volume of patterns generated:** For frequent attribute sets, the number of structural correlation patterns may be large. Most of these patterns are expected to have sizes and densities close to the minimum size and density thresholds, respectively. On the other hand, in practice, the most interesting patterns are the largest and densest ones.

Algorithm 12 identifies the top-k structural correlation patterns induced by an attribute set S . It receives the induced graph $\mathcal{G}(S)$, the value of k , and the dense subgraph parameters γ_{min} and min_size . The algorithm returns the top-k structural correlation patterns in terms of size and density. The size is the primary criterion. If two patterns have the same size, the density is used as secondary criterion.

We apply a DFS strategy to traverse the set of structural correlation pattern candidates. Differently from the problem of computing the structural correlation, a

BFS strategy does not make sense for discovering structural correlation patterns, since we are interested in the largest patterns. The set of candidate quasi-cliques $qcCands$ is a stack initialized with the pair $(X, candExts(X))$, where $X = \emptyset$ and $candExts$ is the set of vertices from $\mathcal{V}(S)$ that are not removed by the vertex pruning rules described in Section 2.2.1. New pairs $(q.X, q.candExts(X))$ are pushed into and popped out of $qcCands$ until it is empty. If a given pair can produce a pattern larger than the minimum size min_size , this pair is a candidate top-k pattern. The function *candidate-quasi-clique-pruning* prunes vertices from $q.X$ and $q.candExts$. If any vertex from $q.X$ is pruned, this function returns TRUE and the pair is pruned, otherwise it returns FALSE. If $q.X \cup q.candExts(X)$ is a quasi-clique, the algorithm tries to update the set of top-k patterns through the function *try-to-update-top-patterns*. If $q.X \cup q.candExts(X)$ is not a quasi-clique, the algorithm checks whether $q.X$ is a quasi-clique and, if so, it tries to update \mathcal{C} . Moreover, if $q.X \cup q.candExts(X)$ is not a quasi-clique, new quasi-clique candidates are pushed into $qcCands$ by extending $q.X$ (lines 17-21).

Algorithm 13 try-to-update-top-patterns

INPUT: $V, \mathcal{C}, min_size, k$
OUTPUT: min_size

```

1: for all  $q \in \mathcal{C}$  do
2:   if  $V \subseteq q$  then
3:     return
4:   end if
5: end for
6: for all  $q \in \mathcal{C}$  do
7:   if  $q \subseteq V$  then
8:      $\mathcal{C} \leftarrow \mathcal{C} - \{q\}$ 
9:   end if
10: end for
11: if  $|\mathcal{C}| < k$  then
12:    $\mathcal{C} \leftarrow \mathcal{C} \cup \{V\}$ 
13:   if  $|\mathcal{C}| < k$  then
14:     return
15:   else
16:      $q \leftarrow smallest\text{-and-sparsest-pattern}(\mathcal{C})$ 
17:      $min\_size \leftarrow |q|$ 
18:     return
19:   end if
20: end if
21:  $q \leftarrow smallest\text{-and-sparsest-pattern}(\mathcal{C})$ 
22: if  $|V| \geq |q|$  OR  $(|q| = |V|$  AND  $\gamma(q) < \gamma(V))$  then
23:    $\mathcal{C} \leftarrow \mathcal{C} - \{q\}$ 
24:    $\mathcal{C} \leftarrow \mathcal{C} \cup \{V\}$ 
25:    $q \leftarrow smallest\text{-and-sparsest-pattern}(\mathcal{C})$ 
26:    $min\_size \leftarrow |q|$ 
27:   return
28: end if

```

The function *try-to-update-top-patterns*, described by Algorithm 13, manages the insertion of new top-k structural correlation patterns into the set \mathcal{C} . It receives a candidate quasi-clique V to be inserted, the current set of top-k patterns \mathcal{C} , the current minimum size for patterns min_size , and the value of k as parameters and returns an

updated value of min_size . Since structural correlation patterns are maximal, if a candidate pattern V is a subset of any pattern in \mathcal{C} , it is not inserted. For the same reason, any subset of V is removed from \mathcal{C} . The function *smallest-and-sparsest* returns the smallest and sparsest patterns from \mathcal{C} . In case \mathcal{C} has less than k patterns, V is inserted into \mathcal{C} . Otherwise, if V is larger, or has the same size but is denser, than the smallest and sparsest pattern from \mathcal{C} , it replaces such pattern. Moreover, if \mathcal{C} has k patterns, the value of min_size is updated to the size of the smallest pattern from \mathcal{C} .

The top-k structural correlation patterns can be enumerated more efficiently than the complete set of patterns because already known patterns are used to prune the search space for new ones. The effectiveness of this pruning depends on the value of min_size . High values of min_size may enable the pruning of a significant number of candidate top-k patterns. In the next section, we describe the SCPM algorithm, which aggregates the search, pruning, and sampling techniques discussed in the last sections and also applies the algorithm for the identification of the top-k structural correlation patterns proposed in this section.

Algorithm 14 SCPM Algorithm

INPUT: $\mathcal{G}, \sigma_{min}, \gamma_{min}, min_size, \epsilon_{min}, \delta_{min}, k, \theta_{max}$
OUTPUT: \mathcal{P}

```

1:  $\mathcal{P} \leftarrow \emptyset$ 
2:  $\mathcal{T} \leftarrow \emptyset$ 
3:  $\mathcal{I} \leftarrow frequent\_attributes(\mathcal{G}, \sigma_{min})$ 
4: for all  $S \in \mathcal{I}$  do
5:   if  $\theta_{max} > 0$  then
6:     if  $\epsilon_{min} > \delta_{min} \cdot \epsilon_{exp}(\sigma(S))$  then
7:        $\epsilon \leftarrow structural\_correlation\_with\_sampling(S, \mathcal{G}(S), \gamma_{min}, min\_size, \epsilon_{min} \cdot \sigma(S), \theta_{max})$ 
8:     else
9:        $\epsilon \leftarrow structural\_correlation\_with\_sampling(S, \mathcal{G}(S), \gamma_{min}, min\_size, \delta_{min} \cdot \epsilon_{exp}(S) \cdot \sigma(S), \theta_{max})$ 
10:    end if
11:  else
12:    if  $\epsilon_{min} > \delta_{min} \cdot \epsilon_{exp}(\sigma(S))$  then
13:       $\epsilon \leftarrow structural\_correlation(\mathcal{G}(S), \gamma_{min}, min\_size, \epsilon_{min} \cdot \sigma(S))$ 
14:    else
15:       $\epsilon \leftarrow structural\_correlation(\mathcal{G}(S), \gamma_{min}, min\_size, \delta_{min} \cdot \epsilon_{exp}(S) \cdot \sigma(S))$ 
16:    end if
17:  end if
18:  if  $\epsilon \geq \epsilon_{min}$  AND  $\epsilon / \epsilon_{exp}(S) \geq \delta_{min}$  then
19:     $\mathcal{Q} \leftarrow top\_k\_structural\_correlation\_patterns(\mathcal{G}(S), k, \gamma_{min}, min\_size)$ 
20:    for all  $q \in \mathcal{Q}$  do
21:       $\mathcal{P} \leftarrow \mathcal{P} \cup (S, q)$ 
22:    end for
23:  end if
24:  if  $\epsilon > 0$  AND  $\epsilon \cdot \sigma(S) \geq \epsilon_{min} \cdot \sigma_{min}$  AND  $\epsilon \cdot \sigma(S) \geq \delta_{min} \cdot \epsilon_{exp}(\sigma_{min}) \cdot \sigma_{min}$  then
25:     $\mathcal{T} \leftarrow \mathcal{T} \cup S$ 
26:  end if
27: end for
28:  $\mathcal{P} \leftarrow \mathcal{P} \cup enumerate\_patterns(\mathcal{T}, \mathcal{G}, \sigma_{min}, \gamma_{min}, min\_size, \epsilon_{min}, \delta_{min}, k, \theta_{max})$ 

```

Algorithm 15 enumerate-patterns

INPUT: $\mathcal{T}, \mathcal{G}, \sigma_{min}, \gamma_{min}, min_size, \epsilon_{min}, \delta_{min}, k, \theta_{max}$
OUTPUT: \mathcal{T}

```

1:  $\mathcal{P} \leftarrow \emptyset$ 
2: for all  $S_i \in \mathcal{T}$  do
3:    $\mathcal{R} \leftarrow \emptyset$ 
4:   for all  $S_j \in \mathcal{T}$  do
5:     if  $i > j$  then
6:        $S \leftarrow S_i \cup S_j$ 
7:       if  $\sigma(S) \geq \sigma_{min}$  then
8:         if  $\theta_{max} > 0$  then
9:           if  $\epsilon_{min} > \delta_{min} \cdot \epsilon_{exp}(\sigma(S))$  then
10:             $\epsilon \leftarrow structural\_correlation\_with\_sampling(S, \mathcal{G}(S), \gamma_{min}, min\_size, \epsilon_{min} \cdot \sigma(S), \theta_{max})$ 
11:          else
12:             $\epsilon \leftarrow structural\_correlation\_with\_sampling(S, \mathcal{G}(S), \gamma_{min}, min\_size, \delta_{min} \cdot \epsilon_{exp}(S) \cdot \sigma(S), \theta_{max})$ 
13:          end if
14:          else
15:            if  $\epsilon_{min} > \delta_{min} \cdot \epsilon_{exp}(\sigma(S))$  then
16:               $\epsilon \leftarrow structural\_correlation(\mathcal{G}(S), \gamma_{min}, min\_size, \epsilon_{min} \cdot \sigma(S))$ 
17:            else
18:               $\epsilon \leftarrow structural\_correlation(\mathcal{G}(S), \gamma_{min}, min\_size, \delta_{min} \cdot \epsilon_{exp}(S) \cdot \sigma(S))$ 
19:            end if
20:          end if
21:          if  $\epsilon \geq \epsilon_{min}$  AND  $\epsilon / \epsilon_{exp}(S) \geq \delta_{min}$  then
22:             $\mathcal{Q} \leftarrow top\_k\_structural\_correlation\_patterns(\mathcal{G}(S), k, \gamma_{min}, min\_size)$ 
23:            for all  $q \in \mathcal{Q}$  do
24:               $\mathcal{P} \leftarrow \mathcal{P} \cup (S, q)$ 
25:            end for
26:          end if
27:          if  $\epsilon > 0$  AND  $\epsilon \cdot \sigma(S) \geq \epsilon_{min} \cdot \sigma_{min}$  AND  $\epsilon \cdot \sigma(S) \geq \delta_{min} \cdot \epsilon_{exp}(\sigma_{min}) \cdot \sigma_{min}$  then
28:             $\mathcal{R} \leftarrow \mathcal{R} \cup S$ 
29:          end if
30:        end if
31:      end if
32:    end for
33:   $\mathcal{P} \leftarrow \mathcal{P} \cup enumerate\_patterns(\mathcal{R}, \mathcal{G}, \sigma_{min}, \gamma_{min}, min\_size, \epsilon_{min}, \delta_{min}, k, \theta_{max})$ 
34: end for

```

4.6 The SCPM Algorithm

At this point, we are able to define an algorithm for structural correlation pattern mining and normalized structural correlation pattern mining. The SCPM (Structural Correlation Pattern Mining) algorithm combines the pruning techniques described in Section 4.3, the search strategies for computing the structural correlation presented in Sections 4.2 and 4.4, and the technique for the identification of the top-k structural correlation patterns discussed in this section. The SCPM algorithm (Algorithm 14) receives the graph \mathcal{G} , the minimum support threshold σ_{min} , the minimum density threshold γ_{min} , the minimum size threshold min_size , the minimum structural correlation threshold ϵ_{min} , the minimum normalized structural correlation threshold δ_{min} , the number of top structural correlation patterns k to be identified, and the maximum margin of error accepted θ_{max} as parameters. It gives as output the set of structural correlation patterns (S, V) from \mathcal{G} such that $\sigma(S) \geq \sigma_{min}$, $\epsilon(S) \geq \epsilon_{min}$, $\delta(S) \geq \delta_{min}$, the margin of error of the structural correlation is limited to θ_{max} , $V \in \mathcal{V}(S)$, $|V| \geq min_size$,

$\gamma(V) \geq \gamma_{min}$ and V is one of the top-k structural correlation patterns induced by S in terms of size and density. Since the SCPM algorithm receives both a minimum structural correlation and a minimum normalized structural correlation, it is able to solve the standard and the normalized version of the structural correlation pattern mining problem.

The initial set of attribute sets is composed by the attributes with a support at least σ_{min} (line 3). If some error is accepted, the structural correlation of attribute sets is computed using sampling (see Algorithm 8), otherwise, the exact structural correlation is computed by the function *structural-correlation* (see Algorithm 5). Both the *structural-correlation* and the *structural-correlation-with-sampling* functions can have their κ_{min} parameter set to $\epsilon_{min} \cdot \sigma(S)$ or $\delta_{min} \cdot \epsilon_{exp}(S) \cdot \sigma(S)$. The highest parameter is selected to maximize pruning. The pruning rules for attribute sets based on ϵ and δ (see Section 4.3) are applied in line 24. Pruned attributes are not included into the set of attributes \mathcal{T} to be extended. The attribute sets that satisfy ϵ_{min} and δ_{min} have their top-k structural correlation patterns identified through the function *top-k-structural-correlation-patterns*. Size one attribute sets are extended by the function *enumerate-patterns*.

Algorithm 15 is a high-level description of the function *enumerate-patterns*. It receives the same set of input parameters of the SCPM algorithm, and also the set of patterns to be extended \mathcal{T} , and returns the set of structural correlation patterns (S, V) that have attribute sets extended from those in \mathcal{T} regarding \mathcal{G} , σ_{min} , ϵ_{min} , δ_{min} , θ_{max} , min_size , γ_{min} and k . New attribute sets are extended through the union of existing ones (line 6). The *enumerate-patterns* function calls itself recursively (line 33) until all valid attribute sets are generated. Attribute sets are enumerated in DFS order, as follows:

$$\{A\}, \{B\}, \{C\}, \{D\}, \{A, B\}, \{A, C\}, \{A, D\}, \{A, B, C\}, \{A, B, D\}, \dots$$

In the next section, we propose parallel algorithms for structural correlation pattern mining. Such algorithms exploit multiple processors in a shared memory environment in order to enable the analysis of large attributed graphs in a feasible time.

4.7 Parallel Algorithms

In the recent years, the availability of computers with multiple processing units has motivated the development of parallel algorithms [Wilkinson and Allen, 2004]. In general terms, a parallel algorithm is able to exploit multiple processing units concurrently as means to achieve high performance. Parallel algorithms are of special interest in data

Algorithm 16 par-structural-correlation

INPUT: $\mathcal{G}(S)$, γ_{min} , min_size , κ_{min} **OUTPUT:** ϵ

```

1:  $numActiveThreads \leftarrow 0$ 
2:  $X \leftarrow \emptyset$ 
3:  $K \leftarrow \emptyset$ 
4:  $candExts(X) \leftarrow vertex-pruning(\mathcal{V}(S), \gamma_{min}, min\_size)$ 
5: if  $|candExts(X)| \leq \kappa_{min}$  then
6:    $\epsilon \leftarrow -1$ 
7: else
8:    $q \leftarrow (X, candExts(X))$ 
9:   if searchStrategy = BFS then
10:     $globalQCCandsQ.enqueue(q)$ 
11:    for  $t = 1$  to  $numThreads$  do
12:       $par-coverage-BFS(globalQCCandsQ, K, \mathcal{G}(S), \gamma_{min}, min\_size, numActiveThreads, numThreads)$ 
13:    end for
14:   else
15:    if searchStrategy = DFS then
16:       $globalQCCandsS.push(q)$ 
17:      for  $t = 1$  to  $numThreads$  do
18:         $par-coverage-DFS(globalQCCandsS, K, \mathcal{G}(S), \gamma_{min}, min\_size, numActiveThreads, numThreads)$ 
19:      end for
20:    end if
21:   end if
22:    $\epsilon \leftarrow |K|/|\mathcal{G}(S)|$ 
23: end if

```

mining, since they can provide the ability of processing massive datasets and reduce the computing time. In the particular case of graph mining, parallel algorithms has been argued to enable the mining of graph patterns from large graph databases [Buehrer et al., 2006].

This section presents parallel algorithms for structural correlation pattern mining. Along the last sections, we have presented sequential algorithms for computing the structural correlation and identifying top-k structural correlation patterns. We parallelize such algorithms to a shared memory environment. The parallelization strategy applied in all the algorithms is the work pool pattern [Kumar et al., 1994]. More specifically, candidate dense subgraphs to be processed are managed through a global and several local work pools (one per thread), as will be detailed in the next sections.

4.7.1 Computing the Structural Correlation

In Section 4.2, we proposed two strategies for computing the structural correlation of an attribute set. Such strategies differ by the order in which the candidate dense subgraphs are visited (DFS or BFS). This section presents parallel versions of these algorithms.

Algorithm 16 is a high-level description of a parallel algorithm for computing the structural correlation. It receives the graph $\mathcal{G}(S)$, the minimum density γ_{min} , the minimum size min_size , and the minimum structural coverage κ_{min} as parameters and

Algorithm 17 par-coverage-BFS**INPUT:** $globalQCCands$, K , $\mathcal{G}(S)$, γ_{min} , min_size , $numActiveThreads$, $numThreads$

```

1: while TRUE do
2:   lock L1
3:   if  $|qcCands| > 0$  then
4:      $q \leftarrow globalQCCands.dequeue()$ 
5:      $localQCCands.enqueue(q)$ 
6:      $numActiveThreads \leftarrow numActiveThreads + 1$ 
7:   else
8:     if  $numActiveThreads = 0$  then
9:       unlock L1
10:      BREAK
11:    end if
12:  end if
13:  unlock L1
14:  if  $|localQCCands| > 0$  then
15:    while TRUE do
16:       $q \leftarrow localQCCands.dequeue()$ 
17:       $newQCCands \leftarrow \emptyset$ 
18:       $par\_coverage(q.X, q.candExts(X), K, \mathcal{G}(S), \gamma_{min}, min\_size, newQCCands)$ 
19:       $localQCCands.enqueue(newQCCands)$ 
20:      if  $|localQCCands| = 0$  then
21:        lock L1
22:         $numActiveThreads \leftarrow numActiveThreads - 1$ 
23:        unlock L1
24:        BREAK
25:      end if
26:      lock L1
27:      if  $numActiveThreads < numThreads$  AND  $|globalQCCands| = 0$  then
28:         $globalQCCands.enqueue(localQCCands)$ 
29:         $localQCCands \leftarrow \emptyset$ 
30:         $numActiveThreads \leftarrow numActiveThreads - 1$ 
31:        unlock L1
32:      end if
33:      unlock L1
34:    end while
35:  else
36:    SLEEP
37:  end if
38: end while

```

returns the structural correlation of the attribute set S , which induced the graph $\mathcal{G}(S)$. The algorithm applies a BFS or a DFS strategy according to a variable *searchStrategy*. The search strategy determines the data structure and the coverage computation function used by the algorithm. While the BFS uses a queue ($globalQCCandsQ$) and the function *par-coverage-BFS*, the DFS uses a stack ($globalQCCandsS$) and the function *par-coverage-DFS*. The respective coverage computing function is called in parallel for each thread (i.e., execution unit).

The function *par-coverage-BFS*, described by Algorithm 17, receives a queue $globalQCCands$, the coverage K (initially set as empty), the induced graph $\mathcal{G}(S)$, the minimum density γ_{min} , the minimum size min_size , the number of active threads $numActiveThreads$ (initially set to 0), and the number of threads $numThreads$ as parameters. When the function is finished, the set K contains the coverage set from $\mathcal{G}(S)$ with regard to the parameters. Different threads execute the function *par-coverage-BFS*

Algorithm 18 par-coverage-DFS**INPUT:** $globalQCCands$, K , $\mathcal{G}(S)$, γ_{min} , min_size , $numActiveThreads$, $numThreads$

```

1: while TRUE do
2:   lock L1
3:   if  $|qcCands| > 0$  then
4:      $q \leftarrow globalQCCands.pop()$ 
5:      $localQCCands.push(q)$ 
6:      $numActiveThreads \leftarrow numActiveThreads + 1$ 
7:   else
8:     if  $numActiveThreads = 0$  then
9:       unlock L1
10:      BREAK
11:    end if
12:  end if
13:  unlock L1
14:  if  $|localQCCands| > 0$  then
15:    while TRUE do
16:       $q \leftarrow localQCCands.pop()$ 
17:       $newQCCands \leftarrow \emptyset$ 
18:       $par\_coverage(q.X, q.candExts(X), K, \mathcal{G}(S), \gamma_{min}, min\_size, newQCCands)$ 
19:       $localQCCands.push(newQCCands)$ 
20:      if  $|localQCCands| = 0$  then
21:        lock L1
22:         $numActiveThreads \leftarrow numActiveThreads - 1$ 
23:        unlock L1
24:        BREAK
25:      end if
26:      lock L1
27:      if  $numActiveThreads < numThreads$  AND  $|globalQCCands| = 0$  then
28:         $globalQCCands.push(localQCCands)$ 
29:         $localQCCands \leftarrow \emptyset$ 
30:         $numActiveThreads \leftarrow numActiveThreads - 1$ 
31:        unlock L1
32:      end if
33:      unlock L1
34:    end while
35:  else
36:    SLEEP
37:  end if
38: end while

```

concurrently.

Each thread has the access to the shared queue $globalQCCands$ and the variable $numActiveThreads$ controlled by a lock variable **L1**. Dense subgraph candidates $(q.X, q.candExts(X))$ are dequeued from $globalQCCands$ and enqueued into $localQCCands$, which is the local queue of a thread. Candidates from the local queue are processed through the function *par-coverage* (Algorithm 19) that updates the coverage K and returns a new set of dense subgraph candidates based on the pair $(X, candExts(X))$ received as input. These new pairs are enqueued into the local queue iteratively. Whenever its local queue is empty, a thread becomes inactive and tries to get new pairs from the global queue. If not succeeded, the thread sleeps for some time. A thread that finds another one inactive enqueues its entire local queue into the global queue, becomes inactive, and tries to get a new dense subgraph candidate from the global queue. A thread is finished when the global queue is empty and all the other

Algorithm 19 par-coverage

INPUT: $X, candExts(X), K, \mathcal{G}(S), \gamma_{min}, min_size, newqcCands$

```

1: if candidate-quasi-clique-pruning( $X, candExts(X), \mathcal{G}(S), \gamma_{min}, min\_size$ ) = FALSE then
2:   if is-quasi-clique( $X \cup candExts(X), \gamma_{min}, min\_size$ ) then
3:     lock
4:     for all  $v \in X \cup candExts(X)$  do
5:        $K \leftarrow K \cup v$ 
6:     end for
7:     unlock
8:   else
9:     if is-quasi-clique( $X, \gamma_{min}, min\_size$ ) then
10:      lock L2
11:      for all  $v \in X$  do
12:         $K \leftarrow K \cup v$ 
13:      end for
14:      unlock L2
15:    end if
16:    for all  $v \in candExts(X)$  do
17:       $t.candExts(X) \leftarrow \{u \in q.candExts(X) | u > v\}$ 
18:       $t.X \leftarrow q.X \cup v$ 
19:      lock L2
20:      if  $t.X \cup t.candExts(X) \not\subseteq K$  AND  $|t.X \cup t.candExts(X)| \geq min\_size$  then
21:         $newqcCands \leftarrow newqcCands \cup t$ 
22:      end if
23:      unlock L2
24:    end for
25:  end if
26: end if

```

threads are inactive.

The Algorithm 18 computes the coverage using a DFS strategy. It is very similar to the Algorithm 17 but uses stacks (one local and another global), instead of queues. The function *par-coverage* (Algorithm 19), which is applied by the functions *par-coverage-BFS* and *par-coverage-DFS*, receives the sets X and $candExts(X)$, the minimum size and density of dense subgraphs (γ_{min} and min_size), and the set $newQCCands$ that will contain the extensions of $(X, candExts(X))$ at the end of the execution. A lock variable **L2** restricts the access to the coverage set K to a single thread at time.

The algorithms described along this section compute the structural correlation of attribute sets using multiple threads concurrently as means to achieve high performance. In the next section, we present a parallel algorithm for computing the structural correlation of attribute sets using sampling.

4.7.2 Sampling

This section describes a parallel version of the sampling technique for computing the structural correlation presented in Section 4.4. Instead of computing the structural correlation of the an attribute set S considering the complete induced graph $\mathcal{G}(S)$, we proposed estimating such a correlation using a sample of vertices from $\mathcal{G}(S)$. Therefore,

Algorithm 20 par-check-vertex-in-quasi-clique

INPUT: $v, \mathcal{G}(S), \gamma_{min}, min_size$
OUTPUT: found

```

1:  $globalQCCands \leftarrow \emptyset$ 
2:  $numActiveThreads \leftarrow 0$ 
3:  $X \leftarrow vertex\text{-}pruning(\{v\}, \gamma_{min}, min\_size)$ 
4:  $candExts(X) \leftarrow vertex\text{-}pruning(\mathcal{V}(S), \gamma_{min}, min\_size)$ 
5: if  $X \neq \emptyset$  then
6:    $q \leftarrow (X, candExts(X))$ 
7:   if searchStrategy = BFS then
8:      $globalQCCands.enqueue(q)$ 
9:     for  $t = 1$  to  $numThreads$  do
10:       $par\text{-}find\text{-}quasi\text{-}clique\text{-}BFS(globalQCCands, \mathcal{G}(S), \gamma_{min}, min\_size, found, numActiveThreads,$ 
       $numThreads)$ 
11:    end for
12:   else
13:     if searchStrategy = DFS then
14:        $globalQCCands.push(q)$ 
15:       for  $t = 1$  to  $numThreads$  do
16:         $par\text{-}find\text{-}quasi\text{-}clique\text{-}DFS(globalQCCands, \mathcal{G}(S), \gamma_{min}, min\_size, found, numActiveThreads,$ 
         $numThreads)$ 
17:      end for
18:     end if
19:   end if
20: end if

```

checking whether a vertex is a member of a dense subgraph is a basic operation of the sampling technique for computing the structural correlation. The parallel version of the proposed technique performs this checking concurrently.

The function *par-check-vertex-in-quasi-clique* (Algorithm 20) receives a vertex v , to be checked to be in, at least, one dense subgraph, the induced graph $\mathcal{G}(S)$, and the quasi-clique parameters, γ_{min} and min_size , as parameters (similar to the function *check-vertex-in-quasi-clique*). It returns TRUE if v is in a dense subgraph in $\mathcal{G}(S)$ and FALSE, otherwise. Given a vertex v , it can be verified to be in a dense subgraph using a BFS or a DFS strategy. The BFS strategy, implemented by the function *par-find-quasi-clique-BFS*, traverses the search space of subgraphs using BFS. Similarly, the function *par-find-quasi-clique-DFS* searches for dense subgraphs using DFS. Since the function *par-check-vertex-in-quasi-clique* is very similar to the function *par-structural-correlation* (Algorithm 16), we will not provide much detail on it.

Algorithms 21 and 22 are high-level descriptions of functions *par-find-quasi-clique-BFS* and *par-find-quasi-clique-DFS*, respectively. Function *par-find-quasi-clique-BFS* receives a queue $globalQCCands$, an induced graph $\mathcal{G}(S)$, dense subgraph parameters (γ_{min} and min_size), a boolean variable $found$, the number of active threads $numActiveThreads$, and the number of threads $numThreads$. The variable $found$ is initialized as FALSE and will be set as TRUE in case a quasi-clique is found and FALSE, otherwise. The number of active threads $numActiveThreads$ is initialized as 0. The function *par-find-quasi-clique-DFS* receives the same parameters, except the fact

Algorithm 21 par-find-quasi-clique-BFS

INPUT: *globalQCCands*, $\mathcal{G}(S)$, γ_{min} , *min_size*, *found*, *numActiveThreads*, *numThreads*

```

1: while TRUE do
2:   lock L1
3:   if |qcCands| > 0 AND found = FALSE then
4:     q ← globalQCCands.dequeue()
5:     localQCCands.enqueue(q)
6:     numActiveThreads ← numActiveThreads + 1
7:   else
8:     if numActiveThreads = 0 then
9:       unlock L1
10:      BREAK
11:    end if
12:  end if
13:  unlock L1
14:  if |localQCCands| > 0 then
15:    while TRUE do
16:      q ← localQCCands.dequeue()
17:      newQCCands ← ∅
18:      par-find-quasi-clique(q.X, q.candExts(X),  $\mathcal{G}(S)$ ,  $\gamma_{min}$ , min_size, newQCCands, found)
19:      localQCCands.enqueue(newQCCands)
20:      if |localQCCands| = 0 OR found = TRUE then
21:        lock L1
22:        numActiveThreads ← numActiveThreads - 1
23:        unlock L1
24:        BREAK
25:      end if
26:      lock L1
27:      if numActiveThreads < numThreads AND |globalQCCands| = 0 then
28:        globalQCCands.enqueue(localQCCands)
29:        localQCCands ← ∅
30:        numActiveThreads ← numActiveThreads - 1
31:      end if
32:    end while
33:  end if
34:  else
35:    SLEEP
36:  end if
37: end while

```

that the structure *globalQCCands* is a stack, instead of a queue. In both functions, a lock variable **L1** restricts the access to *globalQCCands* and *numActiveThreads* to a single thread at time.

The function *par-find-quasi-clique*, presented in Algorithm 23, checks whether vertex sets are quasi-cliques and generates new quasi-clique candidates. It is applied by the functions *par-find-quasi-clique-BFS* and *par-find-quasi-clique-DFS*. The variable *found*, shared by multiple threads, has its access managed by the lock variable **L2**. When a dense subgraph is found, the variable *found* is set to TRUE and all threads can finish the search for a dense subgraph.

The algorithms presented in this section compute the structural correlation of an attribute set using sampling and exploiting multiple processors in order to enable the processing of large graphs efficiently. In the next section, we propose a parallel algorithm for the identification of top-k structural correlation patterns in terms of size

Algorithm 22 par-find-quasi-clique-DFS

INPUT: $globalQCCands$, $\mathcal{G}(S)$, γ_{min} , min_size , $found$, $numActiveThreads$, $numThreads$

```

1: while TRUE do
2:   lock L1
3:   if  $|qcCands| > 0$  AND  $found = FALSE$  then
4:      $q \leftarrow globalQCCands.pop()$ 
5:      $localQCCands.push(q)$ 
6:      $numActiveThreads \leftarrow numActiveThreads + 1$ 
7:   else
8:     if  $numActiveThreads = 0$  then
9:       unlock L1
10:      BREAK
11:    end if
12:  end if
13:  unlock L1
14:  if  $|localQCCands| > 0$  then
15:    while TRUE do
16:       $q \leftarrow localQCCands.pop()$ 
17:       $newQCCands \leftarrow \emptyset$ 
18:      par-find-quasi-clique( $q.X, q.candExts(X), \mathcal{G}(S), \gamma_{min}, min\_size, newQCCands, found$ )
19:       $localQCCands.push(newQCCands)$ 
20:      if  $|localQCCands| = 0$  OR  $found = TRUE$  then
21:        lock L1
22:         $numActiveThreads \leftarrow numActiveThreads - 1$ 
23:        unlock L1
24:        BREAK
25:      end if
26:      lock L1
27:      if  $numActiveThreads < numThreads$  AND  $|globalQCCands| = 0$  then
28:         $globalQCCands.push(localQCCands)$ 
29:         $localQCCands \leftarrow \emptyset$ 
30:         $numActiveThreads \leftarrow numActiveThreads - 1$ 
31:        unlock L1
32:      end if
33:      unlock L1
34:    end while
35:  else
36:    SLEEP
37:  end if
38: end while

```

and density.

4.7.3 Top-k Structural Correlation Patterns

The identification of the top-k, instead of the complete set of structural correlation patterns, was proposed in Section 4.5 with the goals of reducing both the computational cost of enumerating such patterns and the volume of patterns generated to the user. Nevertheless, the search space of top-k structural correlation patterns may still be too large in real settings. Therefore, a parallel algorithm may achieve high performance by searching for top-k structural correlation patterns concurrently.

Algorithm 24 is the pseudocode of the function *par-top-k-structural-correlation-patterns*, which identifies top-k structural correlation patterns from an induced graph. It receives the induced graph $\mathcal{G}(S)$, the value of k , and the dense subgraph parameters,

Algorithm 23 par-find-quasi-clique

INPUT: X , $\text{candExts}(X)$, $\mathcal{G}(S)$, γ_{\min} , min_size , newqcCands , found

```

1: if  $\text{candidate-quasi-clique-pruning}(q.X, q.\text{candExts}(X), \mathcal{G}(S), \gamma_{\min}, \text{min\_size}) = \text{FALSE}$  then
2:   if  $\text{is-quasi-clique}(q.X \cup q.\text{candExts}(X), \gamma_{\min}, \text{min\_size})$  then
3:     lock L2
4:      $\text{found} \leftarrow \text{TRUE}$ 
5:     unlock L2
6:   end if
7:   if  $\text{is-quasi-clique}(q.X, \gamma_{\min}, \text{min\_size})$  then
8:     lock L2
9:      $\text{found} \leftarrow \text{TRUE}$ 
10:    unlock L2
11:   end if
12:   if  $\text{found} = \text{FALSE}$  then
13:     for all  $v \in q.\text{candExts}(X)$  do
14:        $t.\text{candExts}(X) \leftarrow \{u \in q.\text{candExts}(X) \mid u > v\}$ 
15:        $t.X \leftarrow q.X \cup v$ 
16:       if  $|t.X \cup t.\text{candExts}(X)| \geq \text{min\_size}$  then
17:          $\text{newqcCands} \leftarrow \text{newqcCands} \cup t$ 
18:       end if
19:     end for
20:   end if
21: end if

```

Algorithm 24 par-top-k-structural-correlation-patterns

INPUT: $\mathcal{G}(S)$, k , γ_{\min} , min_size

```

1:  $\text{globalQCCands} \leftarrow \emptyset$ 
2:  $\text{numActiveThreads} \leftarrow 0$ 
3:  $X \leftarrow \emptyset$ 
4:  $\text{candExts}(X) \leftarrow \text{vertex-pruning}(\mathcal{V}(S), \gamma_{\min}, \text{min\_size})$ 
5:  $q \leftarrow (X, \text{candExts}(X))$ 
6:  $\text{globalQCCands.push}(q)$ 
7: for  $t = 1$  to  $\text{numThreads}$  do
8:    $\text{par-top-k-scps-thread}(\text{globalQCCands}, \mathcal{G}(S), \gamma_{\min}, \text{min\_size}, k, \mathcal{C}, \mathcal{D}, \text{numActiveThreads}, \text{numThreads})$ 
9: end for
10:  $\mathcal{C} \leftarrow \text{join}(\mathcal{C}, \mathcal{D}, k)$ 
11: return  $\mathcal{C}$ 

```

γ_{\min} and min_size , and returns the top-k structural correlation patterns (\mathcal{C}). Similar to its sequential version, function *par-top-structural-correlation-patterns* applies a DFS strategy in the search for structural correlation patterns.

In Section 4.5 we discussed how the minimum size threshold (min_size) can be updated during the search for top-k patterns based on the a current set of patterns identified, what enables the pruning of candidate patterns that can not be as large as the ones already identified. Since structural correlation patterns are maximal, a new pattern will replace its subsets in the current set of top-k patterns. Nevertheless, while a new pattern can replace only one subpattern from the current set of top-k patterns in the sequential algorithm, a parallel algorithm can allow the replacement of two or more patterns by a new pattern. In such a situation, these subpatterns may increase the value min_size and some valid top-k patterns can be pruned while these subpatterns are not replaced.

We guarantee the correctness of the parallel algorithm for the discovery of the

Algorithm 25 par-top-k-scps-thread

INPUT: $globalQCCands$, $\mathcal{G}(S)$, γ_{min} , min_size , k , \mathcal{C} , \mathcal{D} , $numActiveThreads$, $numThreads$

```

1: while TRUE do
2:   lock L1
3:   if  $|qcCands| > 0$  then
4:      $q \leftarrow globalQCCands.pop()$ 
5:      $localQCCands.push(q)$ 
6:      $numActiveThreads \leftarrow numActiveThreads + 1$ 
7:   else
8:     if  $numActiveThreads = 0$  then
9:       unlock L1
10:      BREAK
11:    end if
12:  end if
13:  unlock L1
14:  if  $|localQCCands| > 0$  then
15:    while TRUE do
16:       $q \leftarrow localQCCands.pop()$ 
17:       $newQCCands \leftarrow \emptyset$ 
18:       $par\_top\_k\_scps\_iteration(q.X, q.candExts(X), \mathcal{C}, \mathcal{D}, \mathcal{G}(S), \gamma_{min}, min\_size, newQCCands, k)$ 
19:       $localQCCands.push(newQCCands)$ 
20:      if  $|localQCCands| = 0$  then
21:        lock L1
22:         $numActiveThreads \leftarrow numActiveThreads - 1$ 
23:        unlock L1
24:        BREAK
25:      end if
26:      lock L1
27:      if  $numActiveThreads < numThreads$  AND  $|globalQCCands| = 0$  then
28:         $globalQCCands.push(localQCCands)$ 
29:         $localQCCands \leftarrow \emptyset$ 
30:         $numActiveThreads \leftarrow numActiveThreads - 1$ 
31:        unlock L1
32:      end if
33:      unlock L1
34:    end while
35:  else
36:    SLEEP
37:  end if
38: end while
39: return  $\mathcal{C}$ 

```

top-k structural correlation patterns by using an auxiliary set \mathcal{D} , instead of only a single set of top-k patterns \mathcal{C} . A structural correlation pattern that can be joined with an existing pattern in \mathcal{C} into a larger pattern is inserted in \mathcal{D} , and not \mathcal{C} . Therefore, a new pattern can not replace more than one pattern from \mathcal{C} . The value of min_size is updated considering only \mathcal{C} and patterns can be moved from \mathcal{D} to \mathcal{C} whenever they do not compromise the results. The resulting set of top-k structural correlation patterns \mathcal{C} is combined with \mathcal{D} through the function *join* (line 10).

The function *par-top-k-scps-thread*, described by Algorithm 25, is executed by the threads concurrently. It is similar to the function *par-coverage-DFS* (Algorithm 18). A global stack $globalQCCands$ is shared by all threads and the access to it and to the number of active threads ($numActiveThreads$) is managed by a lock variable **L1**. Moreover, each thread has its own stack ($localQCCands$). Candidate dense subgraphs

Algorithm 26 par-top-k-scps-iteration

INPUT: $X, \text{candExts}(X), \mathcal{C}, \mathcal{D}, \mathcal{G}(S), \gamma_{\min}, \text{min_size}, \text{newqcCands}, k$

- 1: **lock L2**
- 2: $\text{local_min_size} \leftarrow \text{getmin_size}$
- 3: **unlock L2**
- 4: **if** $|q.X| + |q.\text{candExts}(X)| \geq \text{min_size}$ **then**
- 5: **if** $\text{candidate-quasi-clique-pruning}(X, \text{candExts}(X), \mathcal{G}(S), \gamma_{\min}, \text{local_min_size}) = \text{FALSE}$ **then**
- 6: **if** $|q.X| + |q.\text{candExts}(X)| \geq \text{local_min_size}$ **then**
- 7: **if** $\text{is-quasi-clique}(X \cup \text{candExts}(X), \gamma_{\min}, \text{local_min_size})$ **then**
- 8: **lock L3**
- 9: $\text{min_size} \leftarrow \text{try-to-update-top-patterns}(X \cup \text{candExts}(X), \mathcal{C}, \mathcal{D}, \text{local_min_size})$
- 10: **unlock L3**
- 11: **else**
- 12: **if** $\text{is-quasi-clique}(X, \gamma_{\min}, \text{min_size})$ **then**
- 13: **lock L3**
- 14: $\text{min_size} \leftarrow \text{try-to-update-top-patterns}(X, \mathcal{C}, \mathcal{D}, \text{min_size})$
- 15: **unlock L3**
- 16: **end if**
- 17: **for all** $v \in \text{candExts}(X)$ **do**
- 18: $t.\text{candExts}(X) \leftarrow \{u \in q.\text{candExts}(X) \mid u > v\}$
- 19: $t.X \leftarrow X \cup v$
- 20: $\text{newqcCands.push}(t)$
- 21: **end for**
- 22: **end if**
- 23: **end if**
- 24: **end if**
- 25: **end if**

can be popped from the global to the local stack, or the other way around, depending on whether one of them is empty. At each iteration over the local stack localQCCands , the function $\text{par-top-k-scps-thread-iteration}$ (Algorithm 26) checks whether a candidate vertex set is a structural correlation pattern, updating the set of top-k-structural correlation pattern \mathcal{C} and generating new candidate subgraphs that are pushed into the local stack.

The function $\text{par-try-to-update-top-patterns}$ (Algorithm 27) inserts new top-k patterns into \mathcal{C} . It also updates the value of min_size according to the patterns in \mathcal{C} . The auxiliary set \mathcal{D} stores candidate top-k patterns that can be combined with patterns from \mathcal{C} in the generation of larger patterns. A new top-k pattern V can not be a subset of any pattern in \mathcal{C} or \mathcal{D} . Moreover, subsets of V in \mathcal{C} and \mathcal{D} are removed, since structural correlation patterns are maximal. New patterns are first inserted into \mathcal{D} and, then, the algorithm checks whether patterns from \mathcal{D} can be combined with patterns from \mathcal{C} into larger patterns. This checking is performed by the function $\text{candidate-quasi-clique-pruning}$, which returns TRUE if the vertex set $t \cup q$ can not be extended by vertices from a set $\mathcal{V}(S) - (t \cup q)$ into a quasi-clique for any pattern $q \in \mathcal{C}$. If $t \cup q$ can not produce a larger pattern, then t is removed from \mathcal{D} and inserted into \mathcal{C} . Patterns from \mathcal{D} that do not satisfy the minimum size threshold min_size are also removed. If the size of \mathcal{C} is larger than k , it is reduced by the removal of its smallest and sparsest patterns. The value of min_size returned is updated to the size of the smallest and

Algorithm 27 par-try-to-update-top-patterns

```

INPUT:  $V, \mathcal{C}, \mathcal{D}, min\_size, k$ 
1: for all  $q \in \mathcal{C} \cup \mathcal{D}$  do
2:   if  $V \subseteq q$  then
3:     return  $min\_size$ 
4:   end if
5: end for
6: for all  $q \in \mathcal{C}$  do
7:   if  $q \subseteq V$  then
8:      $\mathcal{C} \leftarrow \mathcal{C} - \{q\}$ 
9:   end if
10: end for
11: for all  $q \in \mathcal{D}$  do
12:   if  $q \subseteq V$  then
13:      $\mathcal{D} \leftarrow \mathcal{D} - \{q\}$ 
14:   end if
15: end for
16:  $\mathcal{D} \leftarrow \mathcal{D} \cup \{V\}$ 
17: for all  $t \in \mathcal{D}$  do
18:   if  $|t| \geq min\_size$  then
19:     for all  $q \in \mathcal{C}$  do
20:       if  $candidate\_quasi\_clique\_pruning(t \cup q, \mathcal{V}(S) - (t \cup q), \mathcal{G}(S), \gamma_{min}, min\_size) = \text{TRUE}$  then
21:          $\mathcal{D} \leftarrow \mathcal{D} - \{t\}$ 
22:          $\mathcal{C} \leftarrow \mathcal{C} \cup \{q\}$ 
23:       end if
24:     end for
25:   else
26:      $\mathcal{D} \leftarrow \mathcal{D} - \{t\}$ 
27:   end if
28: end for
29: while  $|\mathcal{C}| > k$  do
30:    $q \leftarrow smallest\_and\_sparsest\_pattern(\mathcal{C})$ 
31:    $\mathcal{C} \leftarrow \mathcal{C} - \{q\}$ 
32: end while
33: if  $|\mathcal{C}| = k$  then
34:    $\mathcal{C} \leftarrow \mathcal{C} - \{q\}$ 
35:    $\mathcal{C} \leftarrow \mathcal{C} \cup \{V\}$ 
36:    $q \leftarrow smallest\_and\_sparsest\_pattern(\mathcal{C})$ 
37:   return  $|q|$ 
38: end if
39: return  $min\_size$ 

```

sparsest pattern from \mathcal{C} whenever the size of \mathcal{C} is k .

The parallel algorithms presented along this and the previous sections can replace their sequential versions in the SCPM algorithm (Algorithm 14). Therefore, the search, pruning, and sampling techniques can be combined with the parallel algorithms in order to produce an efficient and scalable algorithm.

In this chapter, we have studied the structural correlation pattern mining from an algorithmic perspective. Algorithms for computing the standard and the normalized structural correlation of attribute sets, and for identifying the top- k structural correlation patterns have been proposed and discussed. Pruning, sampling, and parallelization strategies for structural correlation have been applied in order to enable the analysis of large datasets in a feasible time. These algorithms are available as open-source¹. In

¹<http://code.google.com/p/scpm/>

the next chapter, we present an experimental evaluation of such algorithms.

Chapter 5

Experimental Evaluation

This work proposes the analysis of the correlation between attribute sets and dense subgraphs in large attributed graphs. We have argued that such an analysis may provide relevant knowledge regarding the relationship between attributes and the graph topology in real graphs. Moreover, in Chapter 4, we proposed algorithms for structural correlation pattern mining. This chapter presents an extensive experimental evaluation of the structural correlation pattern mining both in terms of the knowledge it provides and the performance of the proposed algorithms.

Section 5.1 presents case studies on the correlation between attribute sets and dense subgraphs. The relevance of the knowledge provided by the structural correlation pattern mining is shown using real attributed graphs from different domains. The datasets used are a collaboration graph, a music social network, a citation graph and a PPI network. We give an overview on the properties of attribute sets and show interesting structural correlation patterns discovered from each dataset. The results are discussed based on the characteristics of the specific application scenarios.

In Section 5.2, we study different input parameters of structural correlation pattern mining algorithms using a real dataset. The idea is to show how these parameters affect the patterns identified by the algorithm, providing guidelines for a proper setting of them.

The performance evaluation of the proposed algorithms is presented in Section 5.3. The main goal of this evaluation is to study the execution time of these algorithms in terms of their input parameters. Along Chapter 3, we gave theoretical bounds for the complexity of algorithms for the structural and the normalized structural correlation pattern mining problems. In this chapter we show how the techniques described in Chapter 4 enable the processing of large graphs in a feasible time.

5.1 Case Studies

In this work, we study the problem of correlating attribute sets and the formation of dense subgraphs in attributed graphs, what we call structural correlation pattern mining. This analysis may provide relevant knowledge in several real-life scenarios. In order to show the applicability of the structural correlation pattern mining, in the next sections, we perform several case studies on the correlation between attribute sets and dense subgraphs in real datasets from different domains.

The datasets used in the case studies are a collaboration graph, a social network, a citation network and a PPI network. In the collaboration graph, vertices are researchers, edges are collaborations and attributes are keywords associated with researchers. In the social network, vertices are people, edges represent friendship and vertex attributes are artists listened to. In the citation network, vertices represent papers, citations are edges, and attributes are terms from papers. We also apply the structural correlation pattern mining to a PPI network, where vertices represent genes, edges represent protein-protein and gene interactions, and attributes are tissues where genes are expressed.

For each dataset, we characterize important aspects related to the structural correlation pattern mining. The degree, attribute frequency and attributes per vertex distributions are plotted in log-log scale, since they are expected to be heavy-tailed. We also enumerate the top structural correlation and normalized structural correlation patterns discovered. For some attribute sets, we show the graphs induced by them.

The expected structural correlation values for different supports using the simulation and analytical models, proposed in Section 3.2, are analyzed for each dataset. The estimates given by the analytical model, which are upper bounds of the expected structural correlation, are compared against the simulation results. For all datasets, we apply the analytical-based normalized structural correlation, given by Definition 8.

The structural correlation patterns presented in the case studies are the largest ones induced by the top structural correlation or normalized structural correlation attribute sets. In some cases, we also show the largest structural correlation patterns discovered overall. Such patterns are analyzed in terms of the specific domain of their respective datasets.

The attribute sets are characterized in terms of the structural coverage, structural correlation, and normalized structural correlation distributions. Such distributions are plotted in semi-log or log-log scale, since they are expected to be heavy-tailed. We also study the correlation among these three functions. Such an analysis may produce insights about the relation among these functions and, as a consequence, assess the

value added by each of them in comparison to the others.

5.1.1 DBLP

In the attributed graph extracted from the DBLP¹ digital library, each vertex represents an author and two authors are connected if they have co-authored a paper. The attributes of authors are terms that appear in the titles of papers authored by them². In the DBLP dataset an attribute set defines a topic (i.e., set of terms that carry a specific meaning in the literature) and a dense subgraph is a community. Therefore, a structural correlation pattern is a community in a given topic. The proposed application is very related to the problems of topic discovery [Pons-Porrata et al., 2007] and expert group search in social networks [Lappas et al., 2009], since a structural correlation pattern defines a group of researchers that have already worked together and have expertise in a given topic. It is important to notice that attributes are considered as a *bag-of-words*. Therefore, some combinations of words may not be semantically valid. A possible solution for this problem would be the use of n-grams, instead of single terms as attributes.

The DBLP dataset contains 108,030 vertices, 276,658 edges, and 23,285 attributes. Figure 5.1 shows the cumulative degree, the attribute frequency and the attributes per vertex distribution for the DBLP dataset. We can notice that such distributions present heavy-tailed behavior. As a consequence, we can expect that few vertices are in large quasi-cliques, few attributes cover a wide set of vertices and few vertices are covered by many structural correlation patterns.

Table 5.1 shows the top 10 attribute sets in terms of structural correlation discovered from the DBLP dataset. The minimum size (*min_size*) and density (γ_{min}) parameters were set to 10 and 0.5, respectively. The minimum support threshold (σ_{min}) was set to 400. The attribute set $\{grid, applic\}$ presents the highest structural correlation (0.26), i.e., 26% of the authors that have the keywords “grid” and “applic” are inside a small community of researchers of size, at least, 10 where each of them have collaborated with half of the other members.

The structural correlation is affected by the support of the attribute sets. The higher is the support, the higher is the expected structural correlation of an attribute set. Figure 5.2, shows the expected structural correlation for different support values in the DBLP dataset. The input parameters are the same used to generate the results shown in Table 5.1. As described in Section 3.2, we propose two models for estimating

¹<http://www.informatik.uni-trier.de/~ley/db>

²The set of attributes was reduced by stemming and removal of stop words.

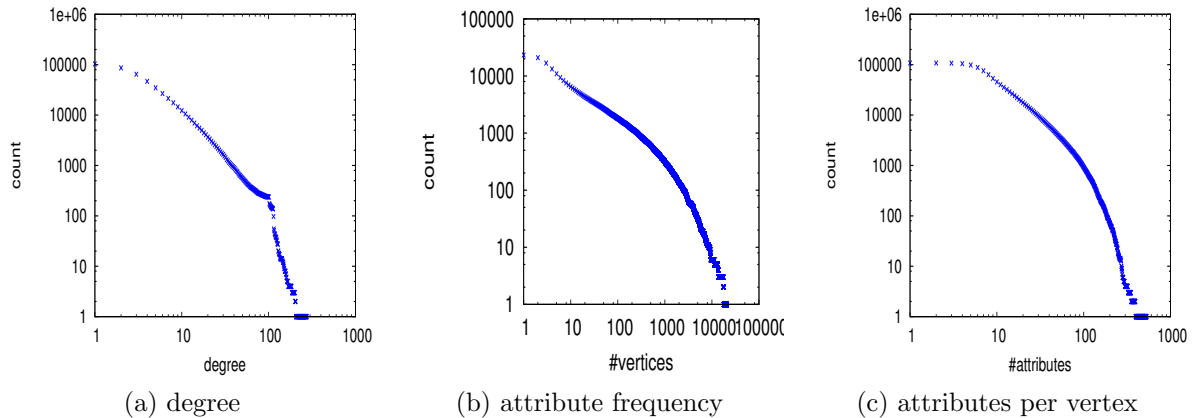


Figure 5.1: Cumulative degree, attribute frequency, and attributes per vertex distribution (in log-log scale) for the DBLP dataset

S	size	σ	κ	ϵ
grid applic	2	840	222	0.26
grid servic	2	599	138	0.23
environ grid	2	525	113	0.21
queri xml	2	615	127	0.21
search web	2	1031	209	0.20
search rank	2	420	81	0.19
dynam simul	2	469	90	0.19
queri data	2	1540	295	0.19
chip system	2	702	132	0.19
data stream	2	1073	198	0.18

Table 5.1: Top- ϵ attribute sets from DBLP

the expected structural correlation: the simulation and the analytical model. For the simulation model, we executed 1000 simulations for each support value and show also the standard deviation of the expected structural correlation estimated. The analytical upper bound is not tight w.r.t. the simulation results, but presents a similar growth, which shows that it enables accurate comparisons between the structural correlation of attribute sets.

Based on the proposed analytical model, Table 5.2 shows the top attribute sets in terms of analytical normalized structural correlation (δ_2). The attribute set $\{search, rank\}$ has the highest normalized structural correlation (635,349), i.e., the structural correlation of this attribute set is 635,349 times the upper bound of its expected structural correlation given by the analytical model. Different from the top ϵ attribute sets, the top δ_2 attribute sets have relatively low support. Figure 5.3 presents the graph induced by $\{search, rank\}$ in the DBLP dataset. Vertices in and out of dense subgraphs are set to distinct colors and shapes. In general, dense subgraphs cover the

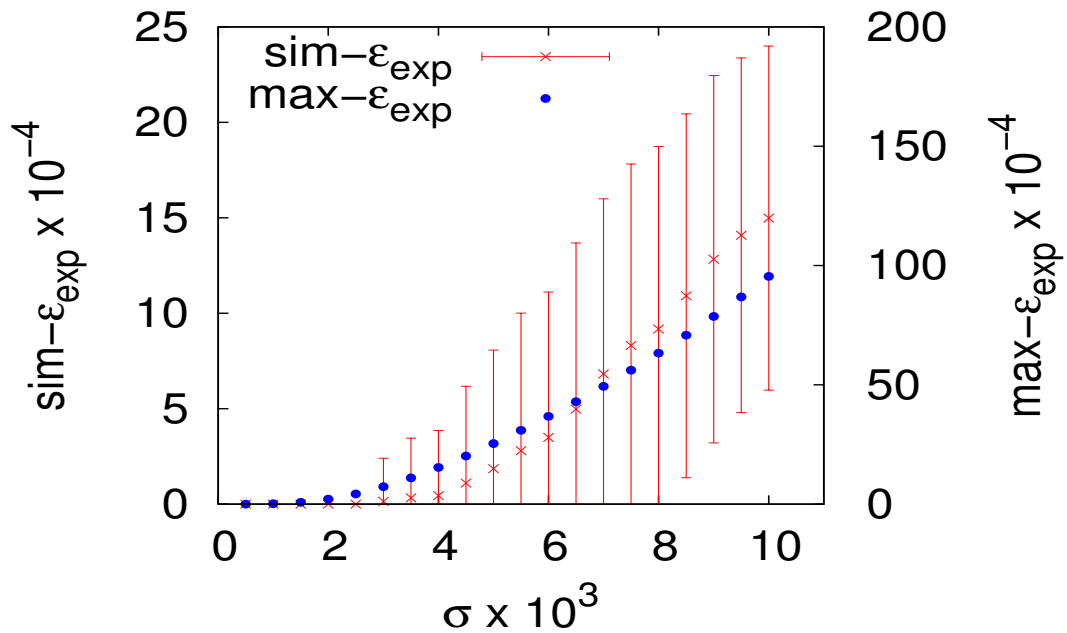


Figure 5.2: Expected structural correlation computed using the simulation model ($\text{sim-}\epsilon_{\text{exp}}$) and the analytical model $\text{max-}\epsilon_{\text{exp}}$

S	size	σ	κ	ϵ	δ_2
search rank	2	420	81	0.19	635,349
perform file	2	404	57	0.14	555,067
structur index	2	404	57	0.14	555,067
search mine	2	413	57	0.14	490,932
us xml	2	400	43	0.11	442,638
search web data	3	424	58	0.14	431,589
base search analysi	3	414	49	0.12	416,385
model internet	2	401	40	0.10	406,059
process data databas	3	416	49	0.12	405,363
perform distribut parallel	3	416	47	0.11	388,818

Table 5.2: Top- δ_2 attribute sets from DBLP

densest components of the induced graph. The concept of structural correlation, defined in this work, assesses how attribute sets induce dense subgraphs in attributed graphs.

The attribute sets shown in Tables 5.1 and 5.2 are related to known research topics in Computer Science. Moreover, it is relevant to analyze the communities induced by keywords with high structural correlation. Figure 5.4 shows a structural correlation pattern induced by the attribute set $\{\text{search}, \text{rank}\}$. Several researchers in the discovered group work, have already worked or have visited Microsoft Research Asia (Jun Yan, Benyu Zhang, Ning Liu, Zheng Chen, Wensi Xi, Weiguo Fan, Wei-Ying Ma,

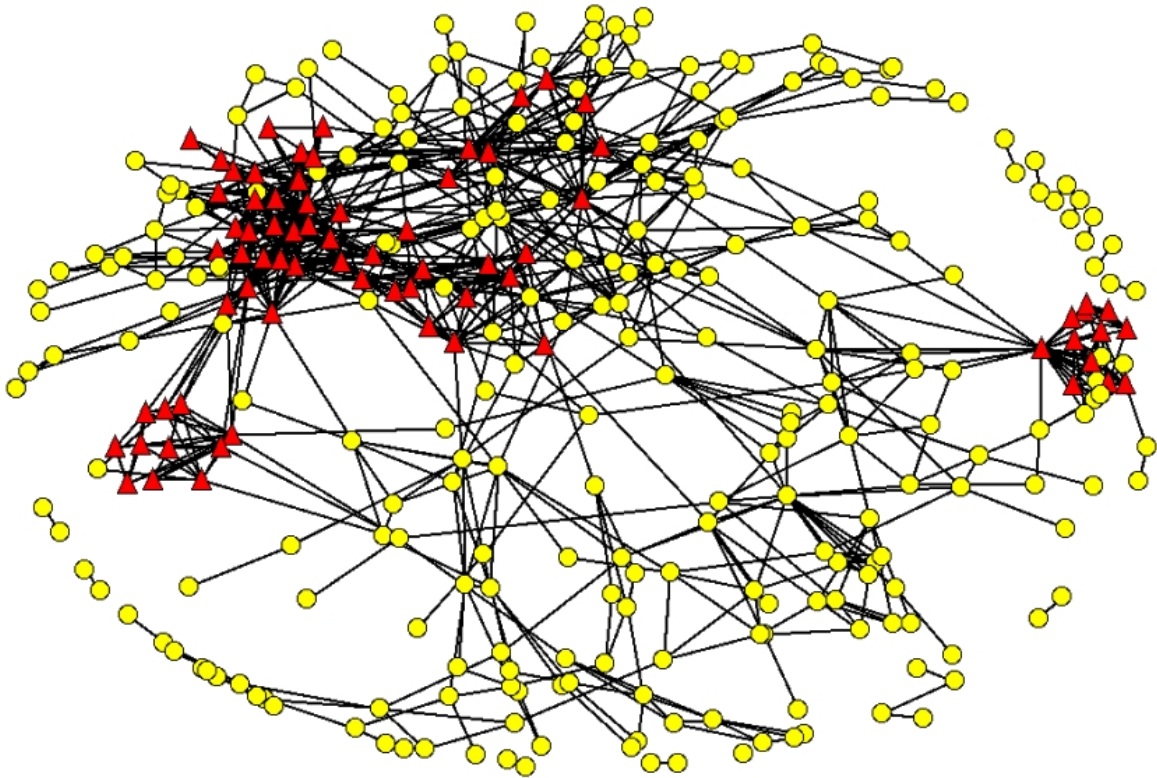


Figure 5.3: Graph induced by the attribute set $\{search, rank\}$ in the DBLP dataset

Shuicheng Yan, Qiang Yang). Weiguo Fan and Edward Fox are professors at Virginia Tech. Wensi Xi got his PhD from Virginia Tech, advised by Weiguo Fan.

The structural correlation pattern shown in Figure 5.5 is the largest pattern found in the DBLP dataset, which is induced by the attribute set $\{perform, system\}$. Most of the researchers in this pattern³ have collaborated while involved in the *Center for Supercomputing Research and Development* at the University of Illinois at Urbana-Champaign. Moreover, the pattern found also includes several members of a project named *Common Runtime Support for High Performance Parallel Languages*⁴, which involved Syracuse University, University of Maryland, Indiana University, University of Rochester, University of Texas (Austin), University of Florida, and Rice University. We checked the homepages of the researchers that compose the discovered pattern and found that they have topics related to systems performance as their interests. Moreover,

³David Padua, Rudolf Eigenmann, Jay Hoeflinger, David Kuck, Pen-Chung Yew, Edward Davidson, Harry Wijshoff, Kyle Gallivan, William Jalby, Allen Maloney, Randall Bramley, Greg Jaxon, Duncan Lawrie, Chuan-Qi Zhu, Jeff Konicek, U Yang, Perry Emrath, Zhiyuan Li, T Murphy, John Andrews, Stephen Turner, Dennis Gannon, Constantine Polychronopoulos

⁴Geoffrey Fox, Sanjay Ranka, Michael Scott, Allen Malony, James Browne, Alok Choudhary, Rudolf Eigenmann, Ian Foster, Dennis Gannon, Tomasz Haupt, Carl Kesselman, Wei Li, Monica Lam, Thomas LeBlanc, David Padua, Constantine Polychronopoulos, Joel Saltz, Alan Sussman, Katherine Yelick

both the research center and the project that motivated the collaborations among the researchers have topics related to systems performance as their research lines.

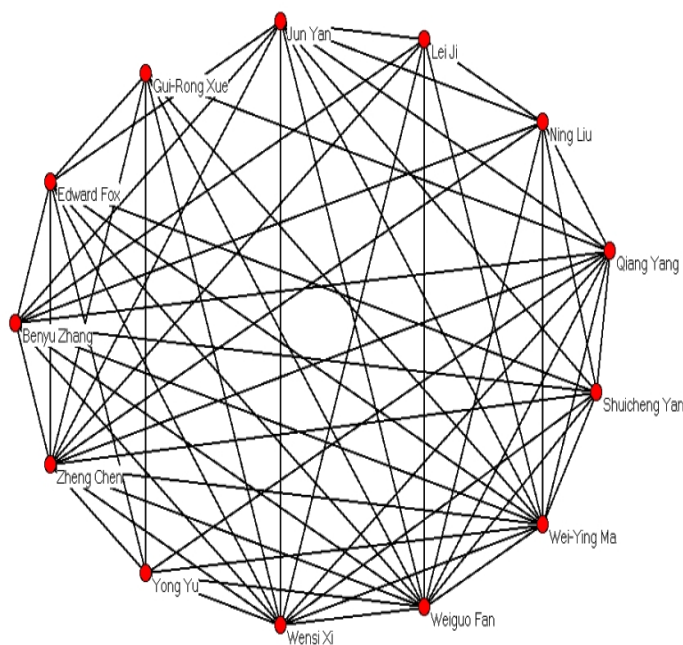


Figure 5.4: Structural correlation pattern induced by the attribute set $\{search, rank\}$, size = 13, and $\gamma=0.58$

An overview of the structural coverage, the structural correlation and the normalized structural correlation in DBLP is shown in Figure 5.6. Figures 5.6a, 5.6b, and 5.6c show the inverse cumulative distribution of the structural coverage, the structural correlation, and the normalized structural correlation, respectively. The three distributions are in semi-log scale and have a slope characteristic of exponential functions. In other words, most of the attribute sets have relatively low structural coverage, structural correlation and normalized structural correlation, while few attribute sets present high values for these functions.

Table 5.3 shows the correlations among the structural coverage, the structural correlation and the normalized structural correlation in DBLP. As expected, smaller attribute sets have higher supports. Moreover, the structural coverage is positively correlated with the support. However, in general, high support attribute sets do not have high structural correlation and the correlation between the support and the normalized structural correlation is negative, i.e., the higher the attribute set support, the lower the normalized structural correlation. Different from the structural correlation, the normalized structural correlation is not biased towards high support attribute sets.

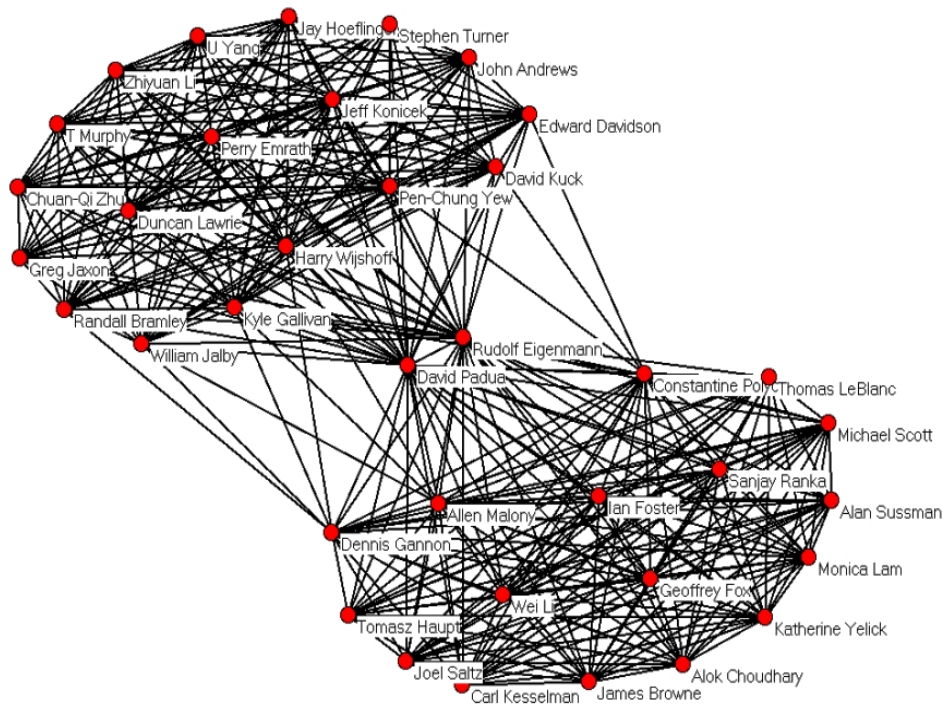


Figure 5.5: Structural correlation pattern induced by the attribute set $\{perform, system\}$, size = 37, and $\gamma=0.5$

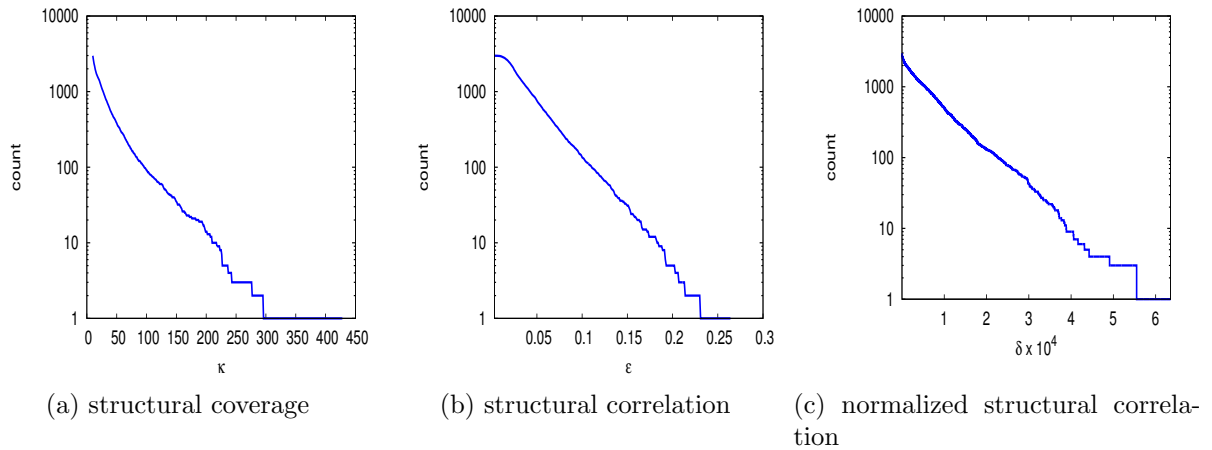


Figure 5.6: Inverse cumulative structural coverage (κ), structural correlation (ϵ), and normalized structural correlation (δ_2)

The structural coverage is also positively correlated with the structural correlation function in DBLP. In other words, the higher is the structural coverage, the higher is the structural correlation of attribute sets. A similar pattern does not occur for the correlation between the structural coverage and the normalized structural correlation. In general, top δ_2 attribute sets have low structural coverage. In particular, larger

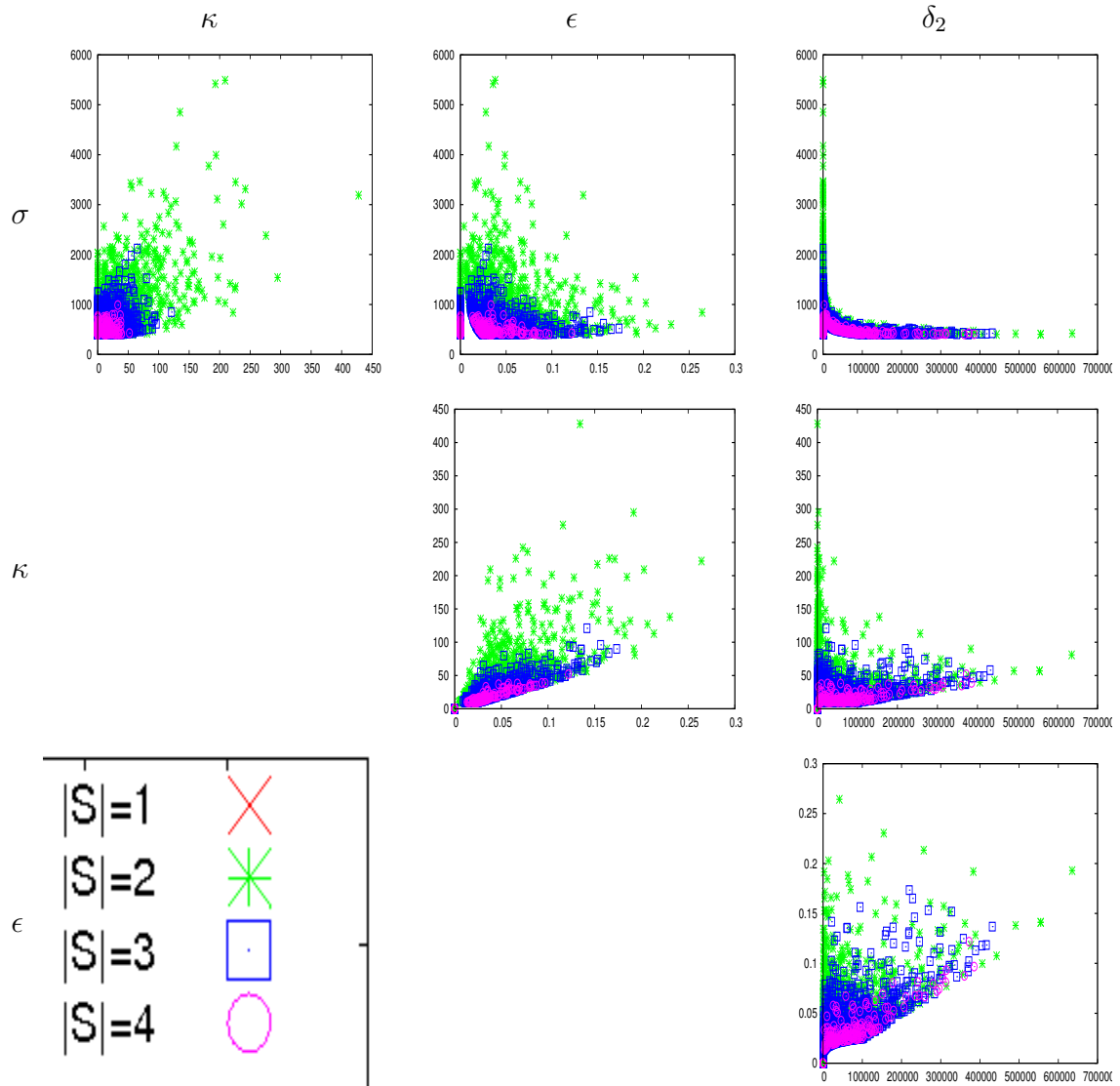


Table 5.3: Scatter plots of the correlations between support (σ), structural coverage (κ), structural correlation (ϵ), and normalized structural correlation for different attribute set sizes ($|S|$) in the DBLP dataset

attribute sets do not have high structural coverage but some of them present high normalized structural correlation.

Since the normalized structural correlation is the ratio of the structural correlation to the expected structural correlation, which is a function that increases with the support, the high normalized structural correlation attribute sets are those with both high structural correlation and low support. Attribute sets with high normalized structural correlation have low support, but the opposite is not necessarily true. The correlations among structural coverage, structural correlation, and normalized structural correlation give a better understanding of the characteristics of each of these

functions. Also, this analysis may support the user in setting the input parameters properly. We are not interested in the semantics behind these correlations.

Many other author attributes could be considered in this study. In particular, it would be interesting to consider location and affiliation as attributes in the identification of interesting correlations between attribute sets and dense subgraphs in collaboration networks [Menezes et al., 2009].

5.1.2 LastFm

LastFm⁵ is an online social music network. In LastFm, users can get connected to their friends and submit song information directly from their music players through a plug-in. Based on this information, LastFm provides several interesting services, such as recommendation, personalized radios, user and artist (singer or group) similarities, among others. We used a sample of the LastFm users crawled through an API provided by the LastFm. In the LastFm network, vertices represent users and edges represent friendships. The attributes of a vertex are the artists the respective user has listened to. An attribute set in the LastFm dataset represents, in a more general interpretation, a musical taste (i.e., set of artists) and a dense subgraph is a community. Structural correlation patterns in the LastFm network are an interesting mechanism to identify musically-oriented communities. Exploring such communities may be of special interest for advertising.

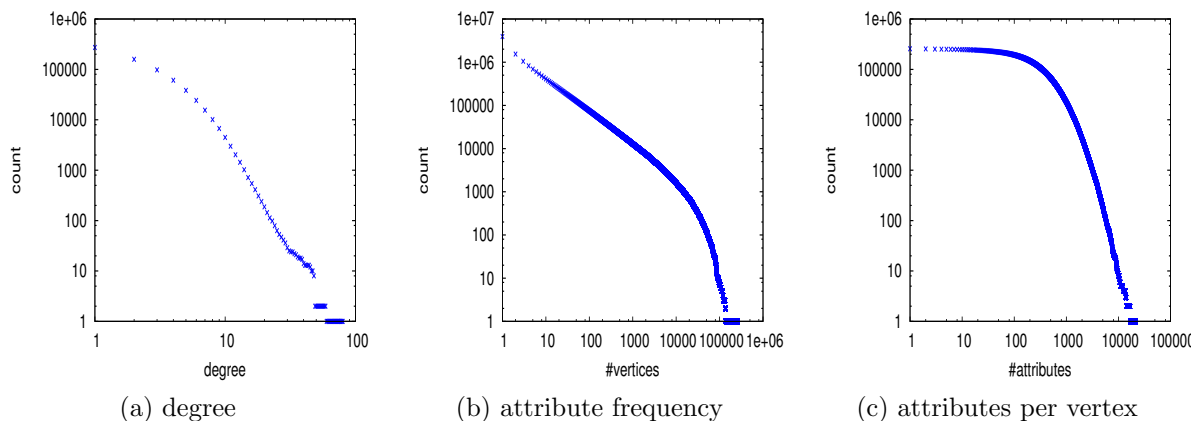


Figure 5.7: Inverse cumulative degree, attribute frequency, and attributes per vertex distribution (in log-log scale) for the LastFm dataset

The LastFm dataset contains 272,412 vertices, 350,239 edges, and 3,929,101 attributes. Figure 5.7 shows the inverse cumulative degree, attribute frequency and

⁵<http://www.last.fm>

attributes per vertex distributions for the LastFm dataset. The three distributions present heavy tails. Only 59% of the users have two or more friends and the user with most friends has 80. Similarly, 50% of the users have listened to less than 230 artists, while the top listener has listened to 21,417 artists. The most popular artist is the band Radiohead (with 121,892 listeners) but most of the artists (61%) have been listened by only one user, what is not a surprise since the artist name is defined by the user and is subject to errors.

S	size	σ	κ	ϵ
Radiohead	1	121,892	13,362	0.11
Coldplay	1	118,053	11,116	0.09
Beatles	1	109,037	10,228	0.09
Red Hot Chili Peppers	1	105,984	9,217	0.09
Metallica	1	83,587	6,405	0.08
Death Cap for Cutie	1	82,025	6,065	0.07
Beck	1	83,360	6103	0.07
Muse	1	94,382	6,792	0.07
Nirvana	1	100,604	7192	0.07
The Shins	1	68,480	4874	0.07

Table 5.4: Top- ϵ attribute sets from the LastFm dataset

Table 5.4 shows the top 10 attribute sets in terms of structural correlation discovered from the LastFm dataset. The minimum size (min_size) and density (γ_{min}) parameters were set to 5 and 0.5, respectively. The minimum support threshold (σ_{min}) was set to 27,000. In general, the top attribute sets are the most frequent ones. The top ϵ attribute set corresponds to the band Radiohead, which is also the most popular band. As discussed in Section 3.2, frequent attribute sets are more likely to cover dense subgraphs than the low frequency ones. However, if we consider the normalized structural correlation, which takes into account the expected structural correlation of an attribute set, the top patterns change significantly. Figure 5.8 shows the expected structural correlation for support values varying from 20,000 to 100,000. Each simulation-based expected structural correlation value corresponds to an average from 1000 simulations. Table 5.5 shows the top normalized structural correlation attribute sets from the LastFm dataset. The top δ_2 attribute set {Sufjan Stevens, Wilco} includes the American singer and songwriter Sufjan Stevens and the American band Wilco. According to LastFm, these two bands have high similarity. Sufjan Stevens is a frequent member of the top δ_2 attribute sets.

Figure 5.9 shows the graph induced by the attribute set {Sufjan Stevens, Wilco}. Vertices inside and outside dense subgraphs are set to different colors and shapes. For clarity, we removed vertices with degree lower than 2. By visualizing vertices inside and

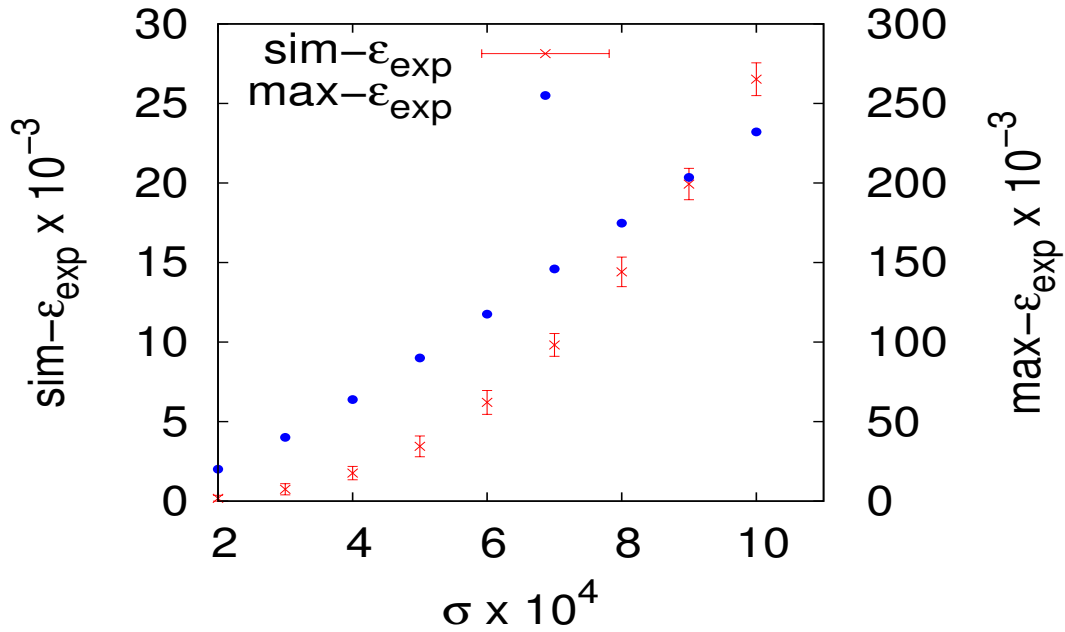


Figure 5.8: Expected structural correlation computed using the simulation model ($\text{sim-}\epsilon_{\text{exp}}$) and the analytical model $\text{max-}\epsilon_{\text{exp}}$

S	size	σ	κ	ϵ	δ_2
Sufjan Stevens, Wilco	2	28,798	1,224	0.04	1.14
Sufjan Stevens, Of Montreal	2	28,621	1,188	0.04	1.13
Beirut	1	27,605	1,067	0.04	1.11
Sufjan Stevens, Decemberists, Beatles	3	27,415	1,046	0.04	1.11
Neutral Milk Hotel, Sufjan Stevens	2	29,260	1,231	0.04	1.10
Sufjan Stevens, Flaming Lips, Beatles	3	27,571	1,048	0.04	1.09
Animal Collective	1	33,555	1,757	0.05	1.09
Broken Social Scene, Neutral Milk Hotel	2	27,308	1,014	0.04	1.09
Radiohead, Spoon, Sufjan Stevens	3	27,113	976	0.04	1.06
Neutral Milk Hotel, Radiohead, Beatles	3	28,776	1,126	0.04	1.04

Table 5.5: Top- δ_2 attribute sets from the LastFm dataset

outside structural correlation patterns, we can understand how the structural correlation captures the relationship between attributes and the formation of dense subgraphs. An example of a structural correlation pattern induced by {Sufjan Stevens, Wilco} is shown in Figure 5.10. We omit the user names due to privacy issues. The largest structural correlation pattern found in the LastFm dataset is presented in Figure 5.11. It represents a community of 34 users who have listened to the Northern Irish singer and songwriter Van Morrison. Finding such a large and dense community in a graph where only 59% of the users have at least two friends is a very interesting result. By correlating artists and the emergence of communities in Lastfm, we can add semantic

value to these communities.

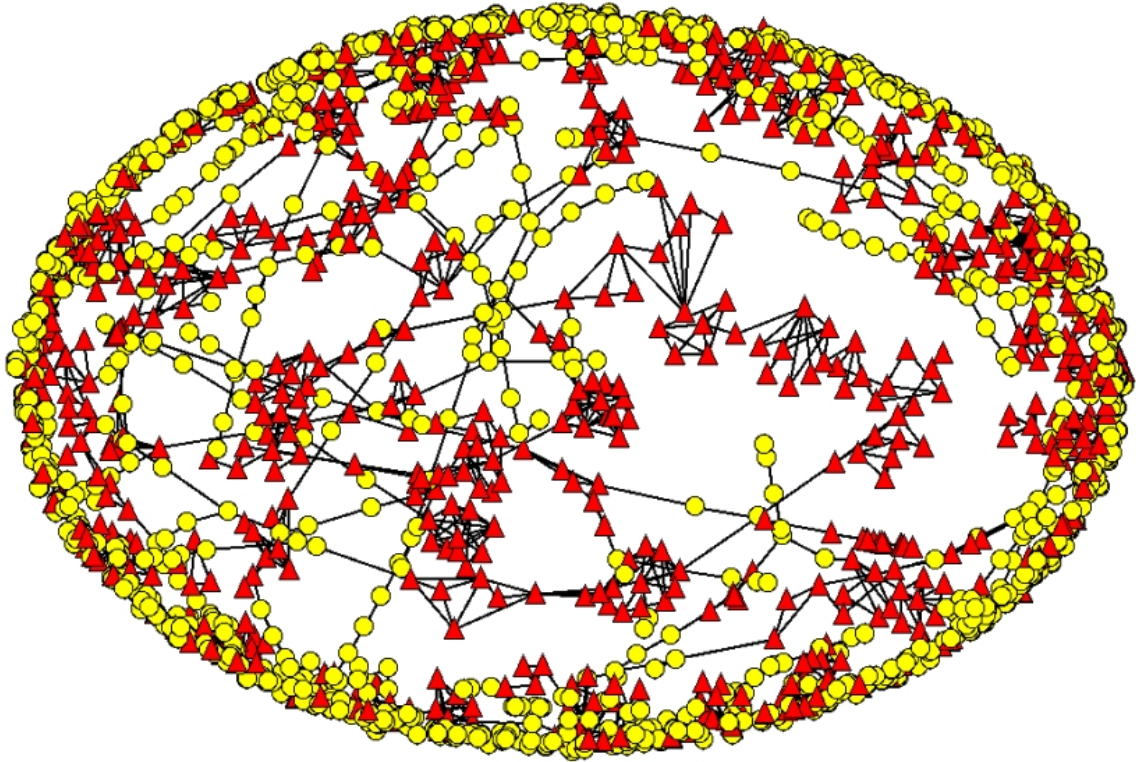


Figure 5.9: Graph induced by the attribute set {Sufjan Stevens, Wilco} in the LastFm dataset

A general view of the properties of the attribute sets from LastFm is shown in Figure 5.12. The inverse cumulative distribution of the structural coverage (log-log scale), the structural correlation (semi-log scale), and the normalized structural correlation (semi-log scale) are presented in Figures 5.12a, 5.12b, and 5.12c, respectively. The shape of the distributions, in their respective scales, show that while the structural coverage may follow a power-law distribution, the structural correlation and the normalized structural correlation are expected to follow an exponential distribution.

We also study the correlations between the different functions for the analysis of attribute sets discussed in this work: the support (σ), the structural coverage (κ), the structural correlation (ϵ), and the normalized structural correlation (δ_2). Table 5.6 shows the scatter plots of the correlation between each pair of these functions. Each point in a scatter plot corresponds to an attribute set, and attribute sets of different sizes (i.e., number of attributes) are distinguished by color. The support of the attribute sets is very correlated with their structural coverage and structural correlation, but not with the normalized structural correlation. Such results are in accord with the previous analysis of the top normalized structural correlation attribute sets. Although

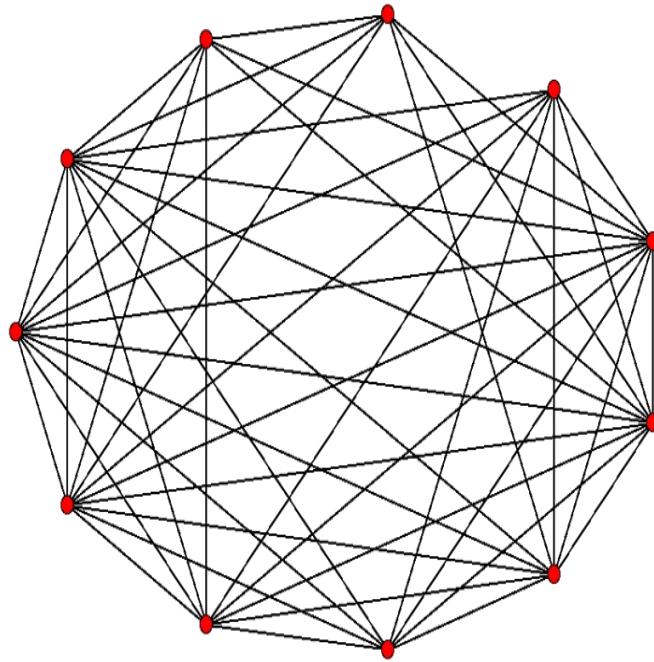


Figure 5.10: Structural correlation pattern induced by the attribute set {Sufjan Stevens, Wilco}, size = 11, and $\gamma=0.7$

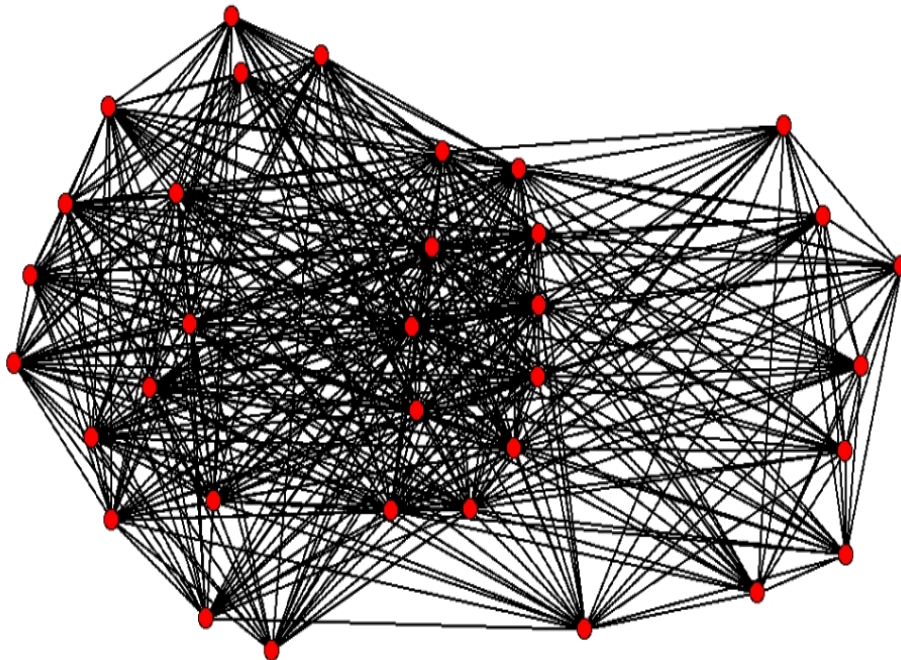


Figure 5.11: Structural correlation pattern induced by the attribute set {Van Morrison}, size = 34, and $\gamma=0.52$

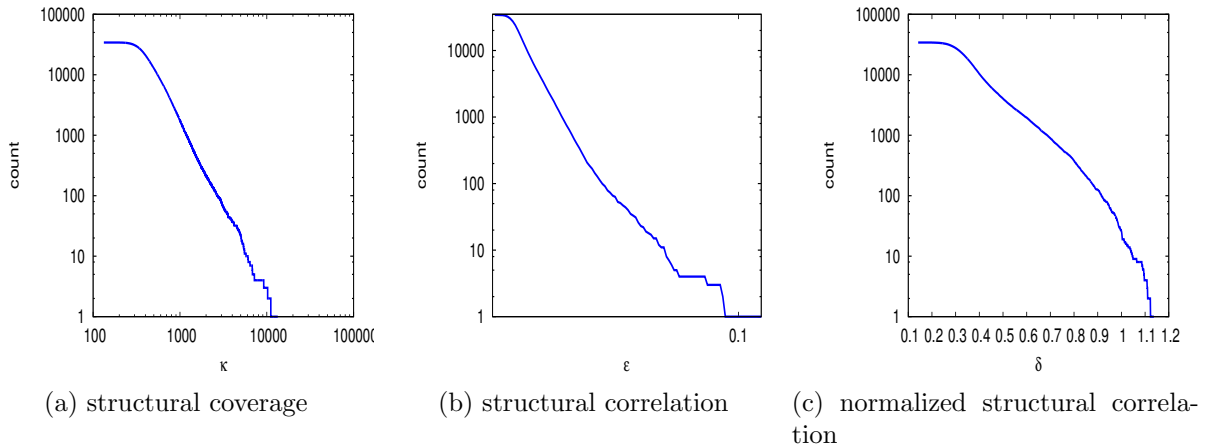


Figure 5.12: Inverse cumulative structural coverage (κ), structural correlation (ϵ), and normalized structural correlation (δ_2)

all the attribute sets evaluated have a correlation with dense communities higher than expected, some less frequent attribute sets present a structural correlation far above the expected.

The low correlation between the attribute set support and its normalized structural correlation deserves a deeper discussion. Considering single artists, it is not surprising that very popular artists do not have a structural correlation much higher than expected. The low degree of most of the users in the LastFm graph (see Figure 5.7a) turns it difficult to an artist to become very popular without the “help” of low degree users. In other words, the graph topology acts as a strong limitation for the normalized structural correlation of popular artists. Moreover, the structural correlation of attribute sets is related to social influence and homophily [Anagnostopoulos et al., 2008], two important social patterns, and less popular artists are expected to depend more on such social patterns. Popular artists have access to other means to maintain and increase their popularity, such as tv appearances and advertisements. In the case of attribute sets that combine more than one artist (e.g. {Sufjan Stevens,Wilco} and {Neural Milk Hotel,Radiohead,Beatles}), their low support reduces their expected structural correlation. However, some of these attribute sets hold a relatively high structural correlation resulting in high values of normalized structural correlation.

The correlation between the structural coverage and the structural correlation function is very clear. The more an attribute set covers dense subgraphs, the higher is its structural correlation. Nevertheless, similar to the support, high values of structural coverage do not necessarily lead to high normalized structural correlations. The correlation between the structural correlation and the normalized structural correlation

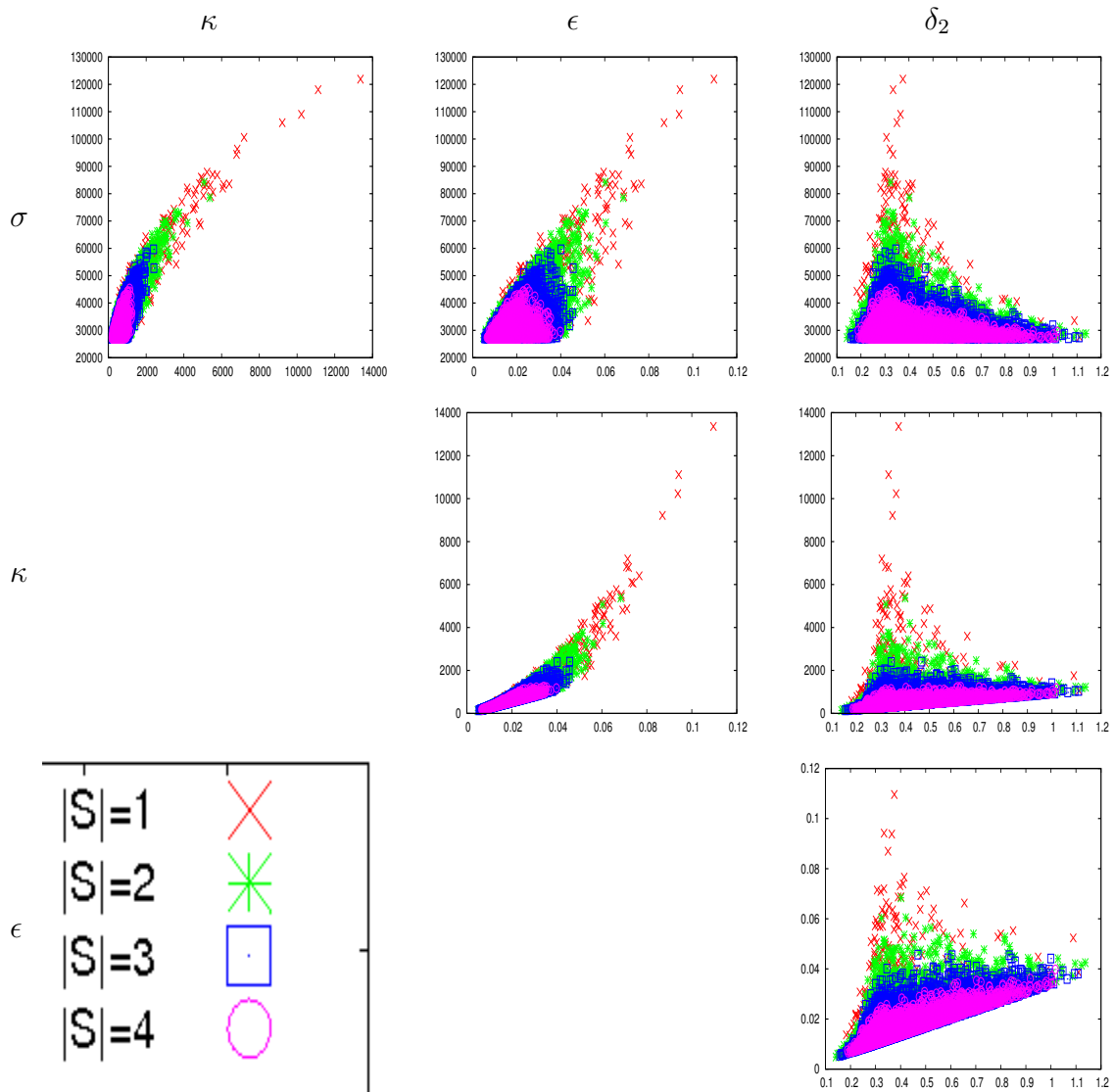


Table 5.6: Scatter plots of the correlations between support (σ), structural coverage (κ), structural correlation (ϵ), and normalized structural correlation for different attribute set sizes ($|S|$) in the LastFm dataset

gives a better hint of the behavior of the δ_2 function in the LastFm dataset. Top δ_2 attribute sets have an intermediate structural correlation that is significantly higher than expected if we consider their low supports.

5.1.3 CiteSeer

CiteSeerX⁶ is a scientific literature digital library and search engine. The two main topics CiteSeerX focuses on are computer and information sciences. It provides sev-

⁶<http://citeseerx.ist.psu.edu>

eral features such as publication and author search, citation statistics and metadata extraction. We build a citation graph from CiteSeerX as of March of 2010. In the CiteSeer graph, papers are represented by vertices and citations by undirected edges. Each paper has as attributes terms extracted from its abstract⁷. Like in the DBLP graph, attribute sets represent topics in the CiteSeer, however, its dense subgraphs define groups of related work (i.e., a set of papers with several citations among them), instead of communities. This application can be useful in digital libraries, specially in search. Keywords given by a user can be associated to existing topics and sets of structural correlation patterns can be returned by the digital library in order to provide a detailed view on the topic by identifying which sets of papers are closely related (i.e., they are in the same dense subgraph).

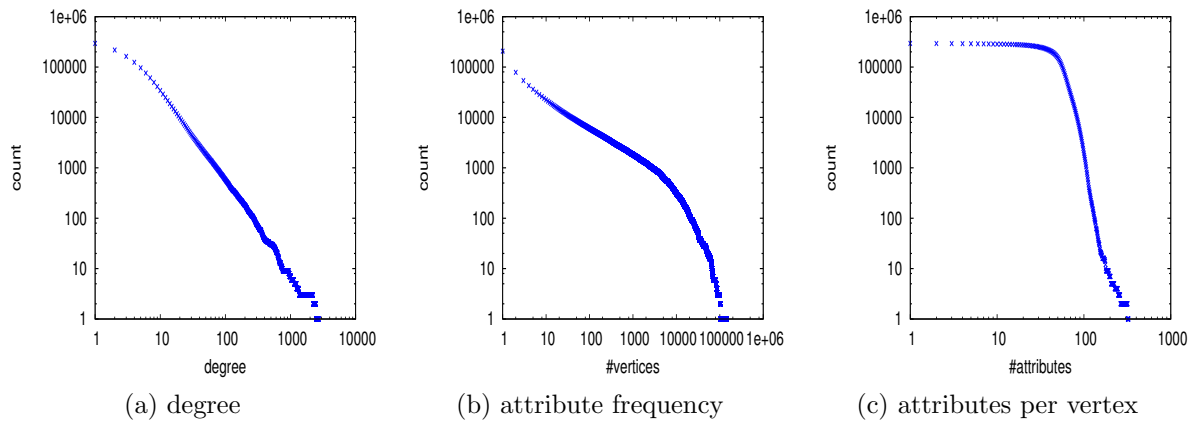


Figure 5.13: Inverse cumulative degree, attribute frequency, and attributes per vertex distributions (in log-log scale) for the CiteSeer dataset

The CiteSeer dataset has 294,104 vertices, 782,147 edges, and 206,430 attributes. Figure 5.13 shows the inverse cumulative degree, attribute frequency and attributes per vertex distributions for the CiteSeer dataset. The vertex degree and attribute frequency distributions are characterized by heavy tails. The attribute per vertex distribution is more homogeneous than the others, since abstract lengths are usually limited by minimum and maximum lengths. Most of the vertices have, at most, 50 attributes and, surprisingly, one vertex has 326 attributes.

The parameters used in the generation of the results for this case study are $\sigma_{min} = 2000$, $min_size = 5$, and $\gamma_{min} = 0.5$. Table 5.7 shows the top structural correlation attribute sets from the CiteSeer dataset. The attribute set $\{network, sensor\}$ has the highest structural correlation (0.47). In general, the top attribute sets are

⁷The set of attributes was reduced by stemming and stop words removal.

related to subject computer networks. We also evaluate the attribute sets from the CiteSeer dataset in terms of the normalized structural correlation. In Figure 5.22, it is shown the expected structural correlation for different support values in CiteSeer. The results for the simulation model are the average expected structural correlation for 1000 simulations.

S	size	σ	κ	ϵ
network sensor	2	3276	1545	0.47
network hoc	2	2744	1279	0.47
ad network hoc	3	2725	1198	0.44
network rout	2	5084	2063	0.41
network wireless	2	5242	2073	0.40
node wireless	2	2086	737	0.35
protocol rout	2	2134	749	0.35
ad network	2	3563	1218	0.34
program logic	2	5895	1939	0.33
memori cach	2	2150	697	0.32

Table 5.7: Top- ϵ attribute sets from CiteSeer

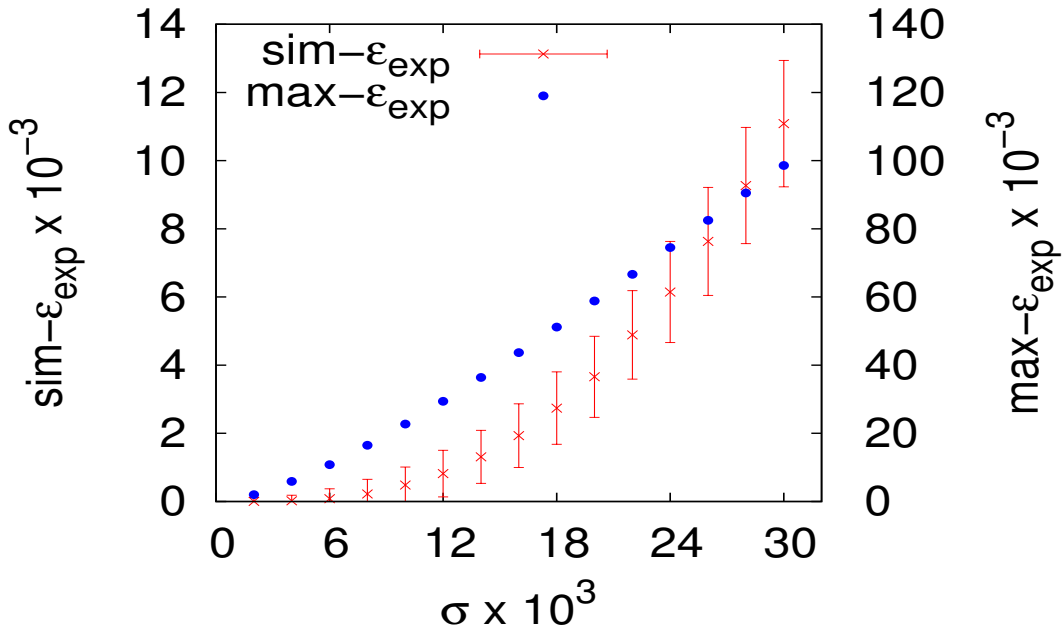


Figure 5.14: Expected structural correlation computed using the simulation model ($\text{sim-}\epsilon_{\text{exp}}$) and the analytical model $\text{max-}\epsilon_{\text{exp}}$

Table 5.8 shows the top normalized structural correlation patterns from CiteSeer. The attribute set $\{\text{node}, \text{wireless}\}$ has the highest normalized structural correlation (164.40). In general, the top δ_2 attribute sets are related to known topics in computer

science. Figure 5.15 shows the graph induced by the attribute set $\{node, wireless\}$ in CiteSeer. Vertices inside and outside dense subgraphs (i.e., in structural correlation patterns) are set to different colors and shapes. We can notice that vertices in dense subgraphs comprehend a densely connected core of the graph, which is surrounded by less connected vertices.

S	size	σ	κ	ϵ	δ_2
node wireless	2	2086	737	0.35	164.40
protocol rout	2	2134	749	0.35	157.57
memori cach	2	2150	697	0.32	143.83
network hoc	2	2744	1279	0.47	141.22
protocol wireless	2	2048	593	0.29	138.70
ad network hoc	3	2725	1198	0.44	134.65
network node rout	3	2075	523	0.25	118.26
optim queri	2	2094	535	0.26	118.17
perform instruct	2	2111	536	0.25	115.95
paper ad network	3	2081	485	0.23	108.86

Table 5.8: Top- δ_2 attribute sets from the CiteSeer dataset

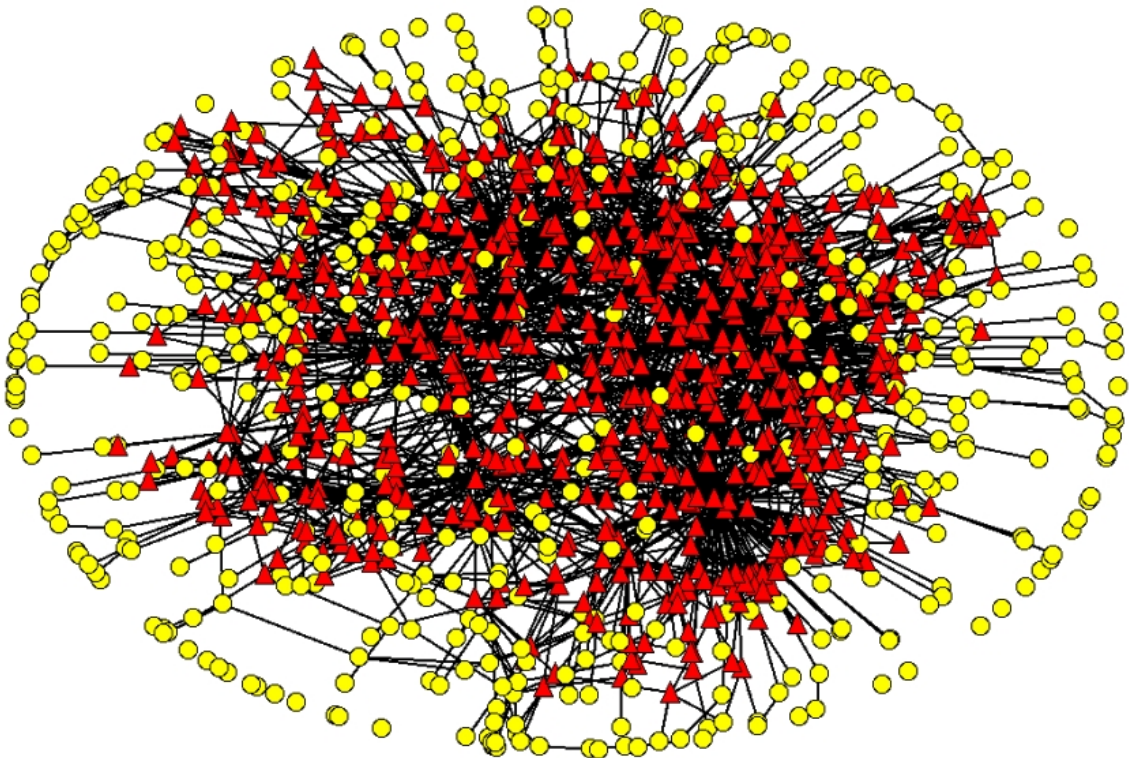


Figure 5.15: Graph induced by the attribute set $\{node, wireless\}$ in the CiteSeer dataset

A particular structural correlation pattern induced by the attribute set $\{node, wireless\}$ is shown in Figure 5.16. By checking the titles of the papers rep-

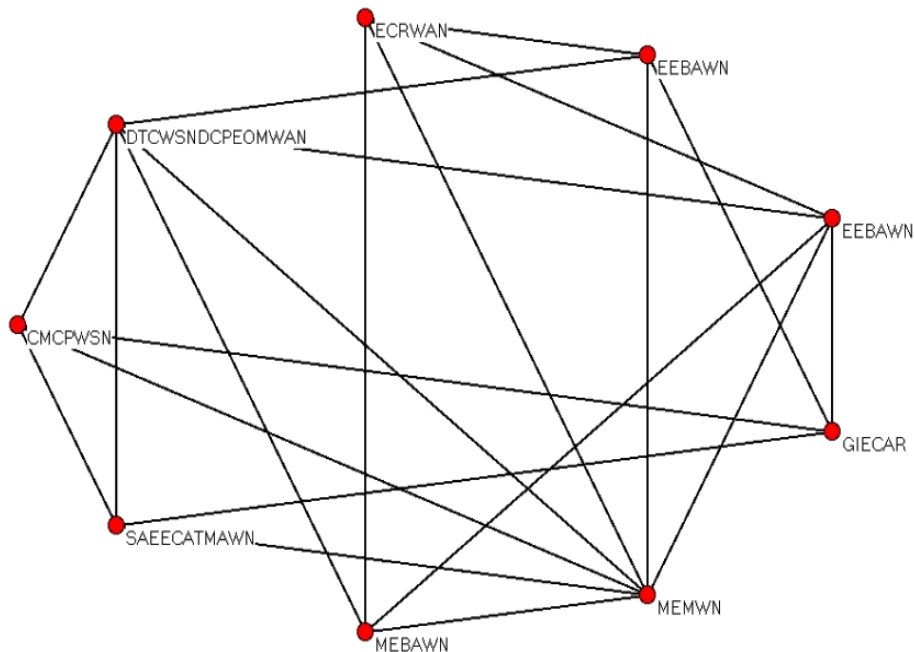


Figure 5.16: Structural correlation pattern induced by the attribute set $\{node, wireless\}$, size = 9, and $\gamma=0.5$ (EEBAWN - Energy-efficient Broadcasting in All-wireless networks, SAEECATMAWN - Span: An Energy-efficient Coordination Algorithm for Topology Maintenance in Ad-hoc Wireless Networks, ECRWAN - Energy-conserving Routing in Wireless Ad-hoc Networks, MEMWN - Minimum-Energy Mobile Wireless Networks, CMCPWSN - Connectivity Maintenance and Coverage Preservation in Wireless Sensor Networks, GIECAR - Geography-informed Energy Conservation for Ad-hoc Routing, MEBAWN - Minimum-energy Broadcast in All-wireless Networks, DTCWSNDCPEOMWAN - Distributed Topology Control in Wireless Sensor Networks with Distributed Control for Power-efficient Operation in Multihop Wireless Ad-hoc Networks)

resented by the vertices in the pattern, we can verify that their subjects are related to wireless networks. We have also read the abstract of this papers and confirmed that they are good representatives for the attribute set $\{node, wireless\}$ and its respective related topics. The paper entitled *Energy-efficient Broadcasting in All-wireless networks* appears twice in the pattern due to a duplicate entry for this paper in the CiteSeer dataset.

Figure 5.17 presents the largest structural correlation pattern discovered in the CiteSeer dataset. This pattern was induced by the attribute set $\{perform, system\}$ and is composed by 21 vertices. The papers included in the pattern cover topics such as caching (e.g., Attribute Caches, Tradeoffs in Two-level on-chip Caching), memory management in general (e.g., Memory-system Design Considerations for Dynamically

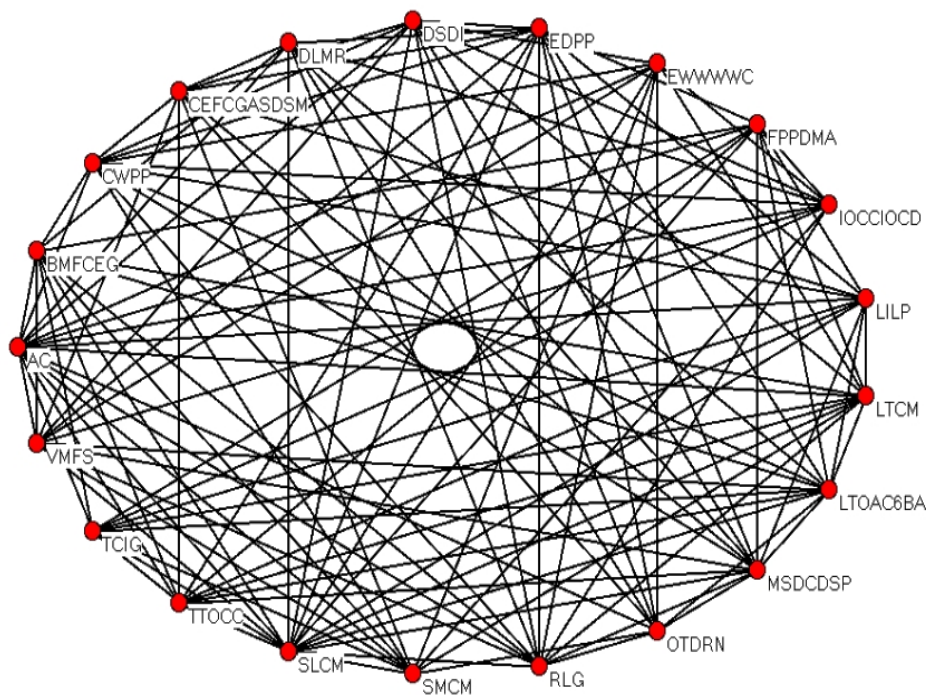


Figure 5.17: Structural correlation pattern induced by the attribute set $\{perform, system\}$, size = 21, and $\gamma=0.5$ (AC - Attribute Caches, SLCM - Systems for Late Code Modification, OTDRN - Observing TCP Dynamics in Real Networks, TCIG - Transparent Controls for Interactive Graphics, LILP - Limits of Instruction Level Parallelism, DLMR - DECWRL/Livermore Magic Release, LTOAC6BA - Link-time Optimization of Address Calculation on a 64-bit Architecture, LTCM - Link-time Code Modification, MSDCDSP - Memory-system Design Considerations for Dynamically Scheduled Processors, BMFCEG - Boolean Matching for Full-custom ECL Gates, CWPP - Cache Write Policies and Performance, SMCM - Shared Memory Consistency Models, CEFCGASDSM - Comparative Evaluation of Fine and Coarse-grain Approaches for Software Distributed Shared Memory, RLG - Recursive Layout Generation, EDPP - Efficient Dynamic Procedure Placement, DSDI - Drip: A Schematic Drawing Interpreter, FPPDMA - Fluoroelastomer Pressure Pad Design for Microelectronic Applications, TTOCC - Trade-offs in Two-level on-chip Caching, IOCCIOCD - I/O Component Characterization for I/O Cache Designs, VMFS - Virtual Memory vs File System, EWWWWC - Experience with a Wireless World Wide Web Client)

Scheduled Processors, Shared Memory Consistency Models), computer networks (e.g., Observing TCP Dynamics in Real Networks, Experience with a Wireless World Wide Web Client), processor design (e.g., Boolean Matching for Full-custom ECL Gates), and instruction level optimization (e.g., Systems for Late Code Modification, Limits of Instruction Level Parallelism). We found also two papers not related to the topic systems performance in this pattern. The papers entitled *Transparent Controls for Interactive Graphics* and *Fluoroelastomer Pressure Pad Design for Microelectronic Applications*

had their abstract incorrectly extracted by CiteSeerX. Terms describing the laboratory where the respective works were developed, which is a computer systems laboratory, were included in the abstracts.

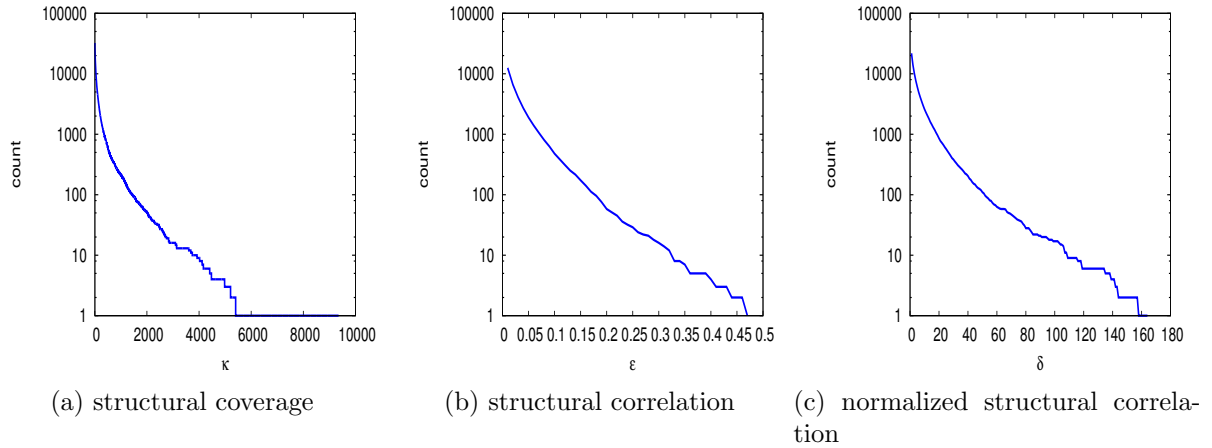


Figure 5.18: Inverse cumulative structural coverage (κ), structural correlation (ϵ), and normalized structural correlation (δ_2)

A general characterization of the attribute sets from CiteSeer in terms of structural coverage, structural correlation, and normalized structural correlation is shown in Figure 5.18. The three distributions are in semi-log scale. We can notice that both the structural coverage, the structural correlation and the normalized structural correlation distributions are heavy-tailed. Similar results were found for the DBLP and the Lastfm dataset.

In Table 5.9, we study the correlations among the structural coverage, the structural correlation and the normalized structural correlation function for attribute sets of different sizes in the CiteSeer dataset. The support of the attribute sets is very correlated with their structural coverage (i.e., attribute sets with high support have, in general, high structural coverage). However, the same does not occur for the structural correlation and the normalized structural correlation functions. High support attribute sets do not necessarily have high structural correlation. Moreover, the top δ_2 attribute sets have low support.

Top structural correlation attribute sets have intermediate support, since high κ attribute sets are usually very frequent and present low structural correlation. Considering the normalized structural correlation, we can notice that it is negatively correlated with the structural coverage. In other words, the higher the structural coverage, the lower is the normalized structural correlation. This property is a consequence of the definition of normalized structural correlation, which considers how the structural

correlation of an attribute set is higher than expected, given its support, the graph topology and the dense subgraph parameters.

High values of normalized structural correlation are usually related to high values of structural correlation. However, the opposite is not true. Attribute sets with high structural correlation but also high support may have low normalized structural correlations, since their structural correlations are not as higher than expected as some attribute sets with high structural correlation and low support.

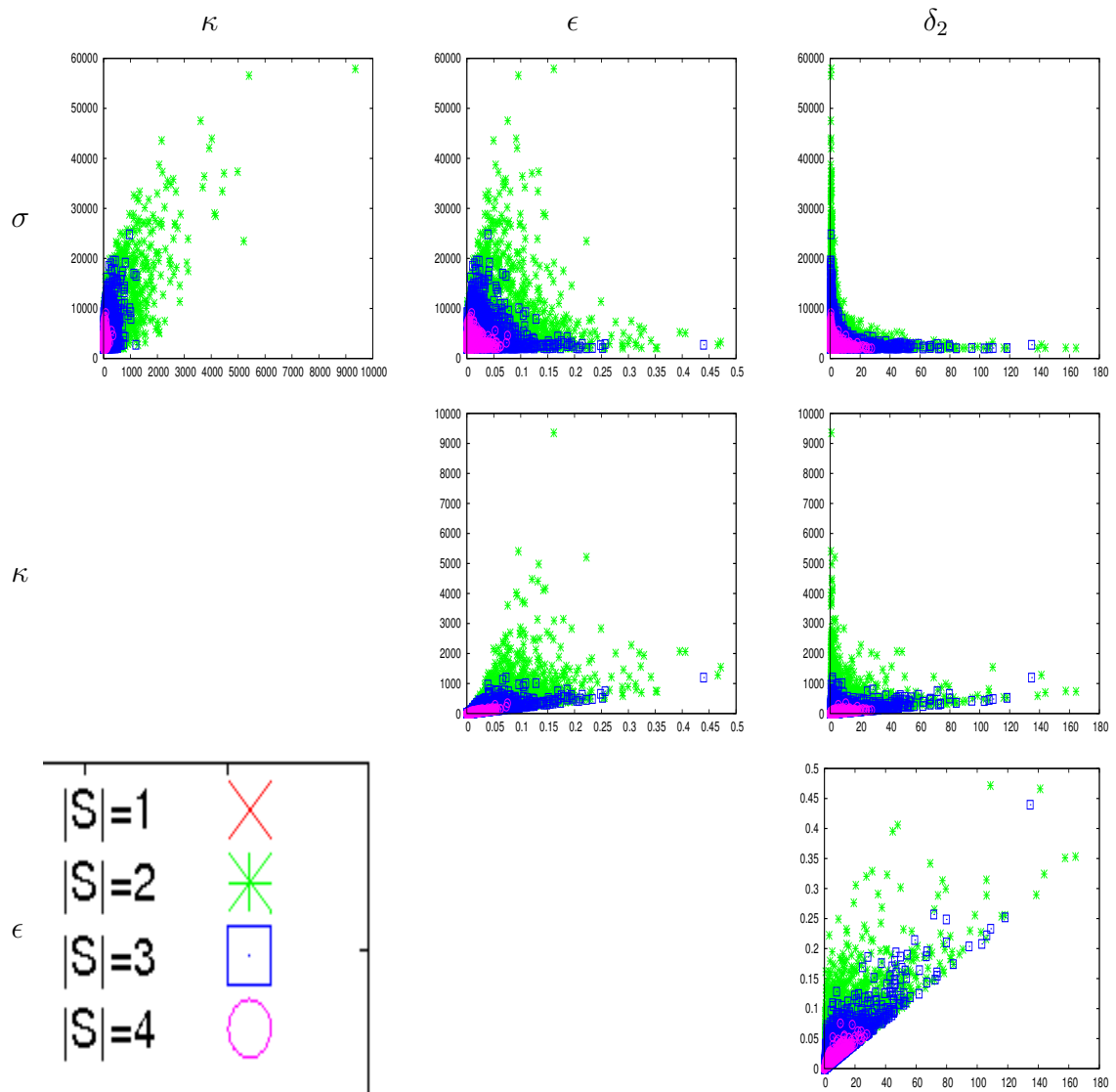


Table 5.9: Scatter plots of the correlations among support (σ), structural coverage (κ), structural correlation (ϵ), and normalized structural correlation for different attribute set sizes ($|S|$) in the CiteSeer dataset

5.1.4 Human

The Human dataset combines a gene network extracted from the BIOGRID database⁸ and a human tissue expression dataset [Shyamsundar et al., 2005]. In a gene network, each vertex is a gene and edges represent protein-protein and genetic interactions. In the specific case of the Human dataset, the genes in the network are from the *Homo sapiens* species. Gene and protein interactions are related to gene biological functions, i.e., genes are connected according to their interactions in biological processes. Gene expression data determines the level of expression of a gene in a set of tissues or cells, which is a result of a microarray experiment [Shyamsundar et al., 2005]. Our dataset is based on the dataset from [Moser et al., 2009].

We create two attributes, corresponding to positive (> 0.5) and negative (< 0.5) normalized values, for each tissue. A gene expressed positively on the tissue SHCN060, for example, will have the attribute +SHCN060. It has been argued that the combined analysis of gene interaction and expression data is more promising than the individual analysis [Moser et al., 2009; Grigoriev, 2001; Hanisch et al., 2002; Ulitsky and Shamir, 2007]. In the Human dataset, an attribute set represents a set of tissues and a dense subgraph represents a *module* [Hartwell et al., 1999]. Therefore, through the analysis of the correlation between attribute sets and the formation of dense subgraphs, we can assess relationships between gene functionality and expression.

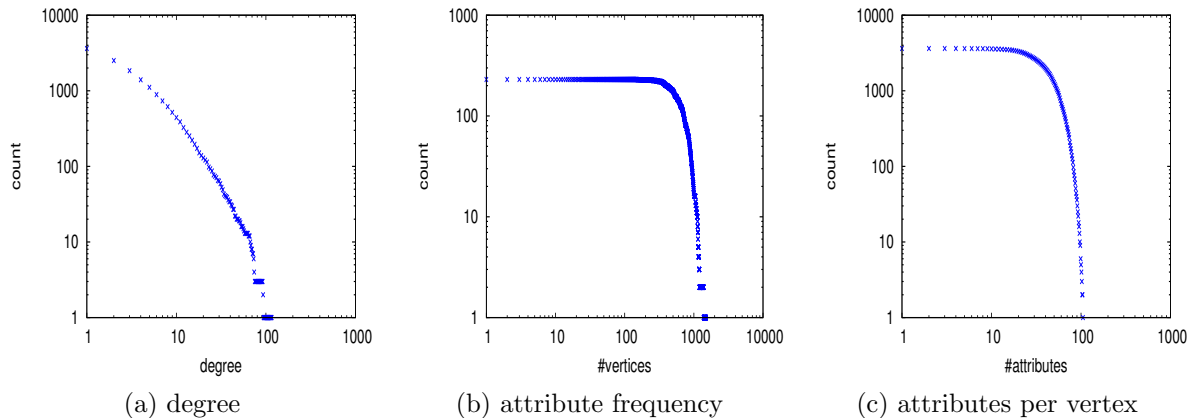


Figure 5.19: Inverse cumulative degree, attribute frequency, and attributes per vertex distributions (in log-log scale) for the Human dataset

The Human dataset contains 3,628 vertices, 8,924 edges, and 230 attributes. Figure 5.19 shows the inverse cumulative degree, attribute frequency and attributes per vertex distributions for the Human dataset. While the degree distribution of the

⁸<http://thebiogrid.org>

genes presents a heavy tailed behavior, the attribute frequency and the attribute per vertex distributions present considerably high minimum values followed by heavy tails.

S	size	σ	κ	ϵ
+SHCN086 +SHCN087 +SHCN088	3	328	58	0.18
+SHCN088	1	910	152	0.17
+SHCN078 +SHCN080 +SHCN088	3	1328	54	0.16
+SHCN060 +SHCN078	2	303	48	0.16
+SHCN087 +SHCN088	2	523	81	0.15
+SHCN080 +SHCN088	2	433	67	0.15
+SHCN078 +SHCN087 +SHCN088	3	316	48	0.15
+SHCN080 +SHCN087 +SHCN088	3	343	52	0.15
+SHBW148	1	864	130	0.15
+SHCN060 +SHCN086	2	326	49	0.15

Table 5.10: Top- ϵ attribute sets from the Human dataset

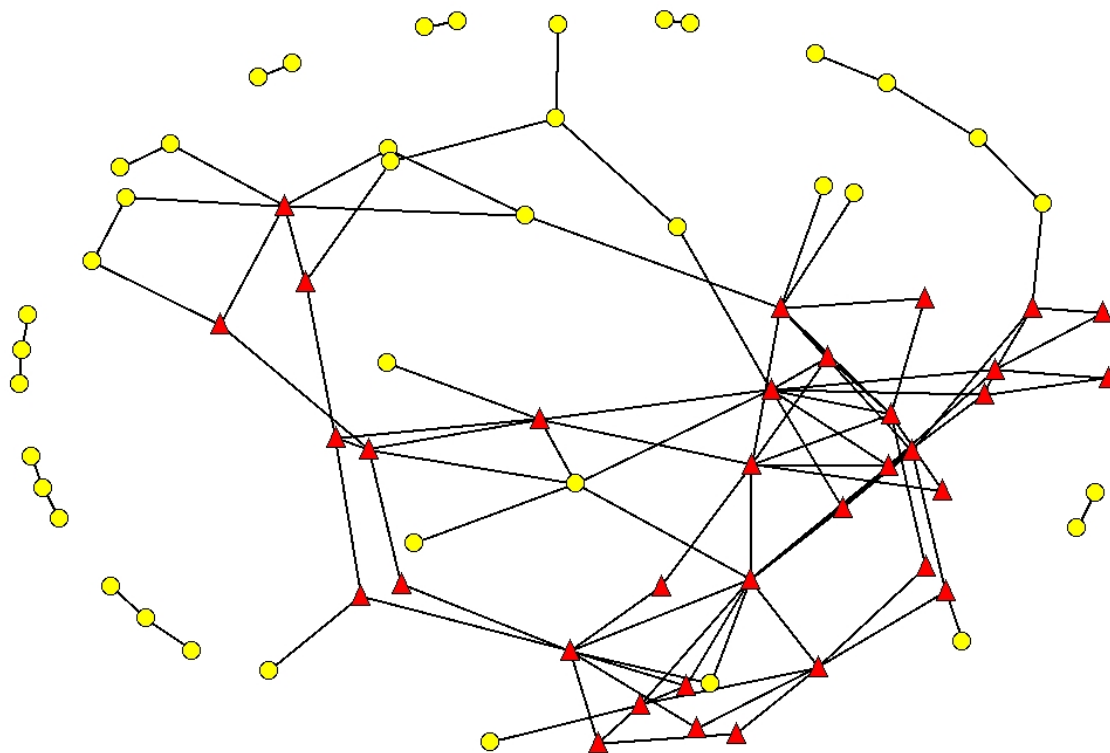


Figure 5.20: Graph induced by the attribute set $\{+SHCN086, +SHCN087, +SHCN088\}$ in the Human dataset

Table 5.10 shows the top 10 attribute sets in terms of structural correlation discovered from the Human dataset. The minimum size (min_size) and density (γ_{min}) parameters were set to 5 and 0.5, respectively. The minimum support threshold (σ) was set to 300. In general, the top attribute sets are composed by attributes corresponding to positive genes in tissues related to the lymphatic system. The lymphatic

system is part of the immune system and includes the tonsil (SHCN078, SHCN086, SHCN088), the thymus (SHCN068, SHBW148), the spleen (SHCN060), and the lymph nodes (SHCN087, SHCN073, SHCN080). The immune system is considered one of the most important vertebrate-specific evolutionary traits [Beck and Habicht, 1996] and it is interesting to notice that genes related to this system present a more cohesive structure. Previous studies have found evidences of the influence of the evolutionary process over PPI networks. In particular, the robustness of such networks to errors [Jeong et al., 2001; Albert et al., 2000; Albert, 2005] and the conservation of cohesive modules across species [Wuchty et al., 2003] may be related to the high structural correlation of immune system-related tissues.

The attribute set $\{+SHCN086, +SHCN087, +SHCN088\}$ presents the highest structural correlation (0.18) in the Human dataset. Figure 5.20 shows the graph induced by such attribute set. Vertices inside and outside structural correlation patterns are set to distinct colors and shapes. Vertices with degree 0 are not shown. We can notice that structural correlation patterns play an important role as components of a dense core of the graph, while the rest of the graph is barely connected.

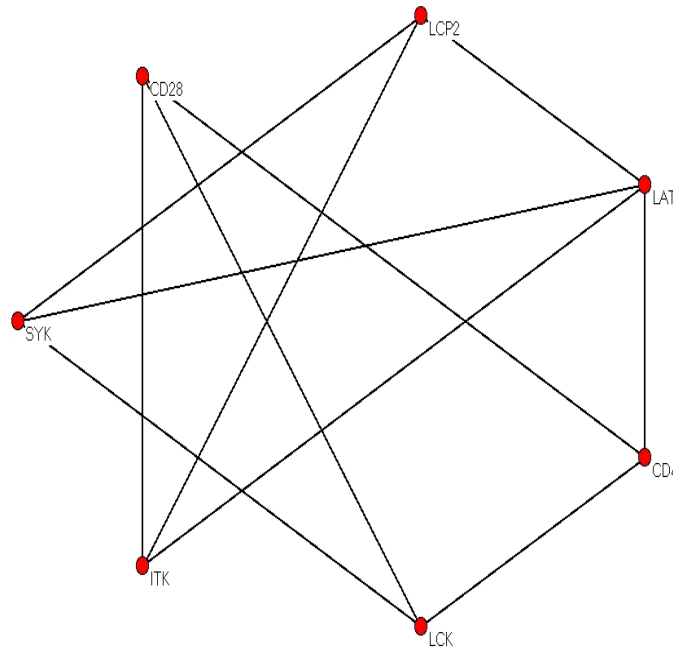


Figure 5.21: Structural correlation pattern induced by the attribute set $\{+SHCN086, +SHCN087, +SHCN088\}$, size = 7, and $\gamma=0.5$

The subgraph shown in Figure 5.21 is a particular structural correlation pattern induced by the attribute set $\{+SHCN086, +SHCN087, +SHCN088\}$. According

to the *Gene Ontology*⁹, the genes CD4, CD28, ITK, LAT, LCK, ITK, and SYK are involved in the immune response process, what explains the occurrence such a structural correlation pattern in a set of tissues related to the lymphatic system.

We also analyze attribute sets from the Human dataset in terms of the normalized structural correlation (see Section 3.2). In the specific case of this dataset, the normalization approach is useful to assess how unexpected is it to find a given correlation between a set of tissues and the existence of modules. Figure 5.22 shows the expected structural correlation ($\gamma=0.5$, $min_size=5$) using the simulation and the analytical models. The simulation results are an average from 1000 executions.

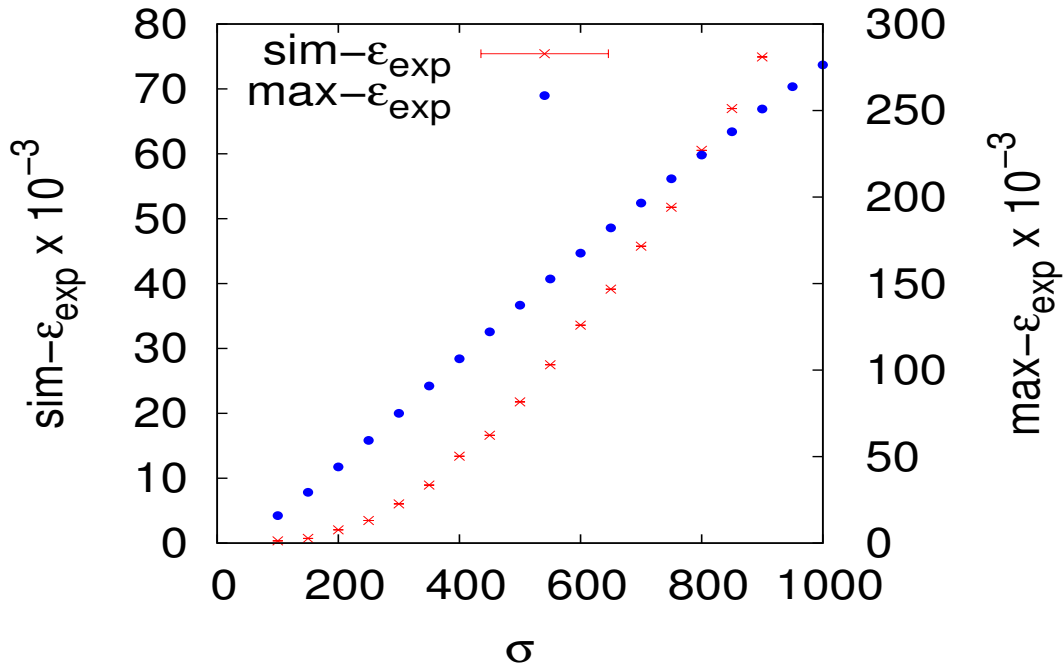


Figure 5.22: Expected structural correlation computed using the simulation model ($sim-\epsilon_{exp}$) and the analytical model $max-\epsilon_{exp}$

Table 5.11 presents the top 10 δ_2 attribute sets from the Human dataset. It is interesting to notice that the top 3 ϵ attribute sets were conserved (see Table 5.10). Moreover, new attribute sets, such as $\{+SHCN068, +SHCN078\}$, were included due to their high structural correlation considering their low supports. Tables 5.10 and 5.11 provide the main properties of the top patterns in terms of structural correlation. However, it is relevant to give a broader view of the attribute sets. Figure 5.23 presents the inverse cumulative structural coverage, structural correlation and normal-

⁹<http://www.geneontology.org>

ized structural correlation distributions for the Human dataset. The distributions are plot in semi-log scale and present heavy-tailed behavior.

S	size	σ	κ	ϵ	δ_2
+SHCN086 +SHCN087 +SHCN088	3	328	58	0.18	2.09
+SHCN060 +SHCN078	2	303	48	0.16	2.08
+SHCN078 +SHCN080 +SHCN088	3	1328	54	0.16	1.96
+SHCN078 +SHCN087 +SHCN88	3	316	48	0.15	1.90
+SHCN068 +SHCN078	2	303	43	0.14	1.86
+SHBW148 +SHCN087 +SHCN088	3	310	45	0.15	1.85
+SHCN060 +SHCN073	2	302	42	0.14	1.84
+SHCN078 +SHCN086 +SHCN087	3	312	45	0.14	1.83
+SHCN60 +SHCN86	2	326	49	0.15	1.80
+SHCN080 +SHCN086 +SHCN088	3	329	49	0.15	1.76

Table 5.11: Top- δ_2 attribute sets from the Human dataset

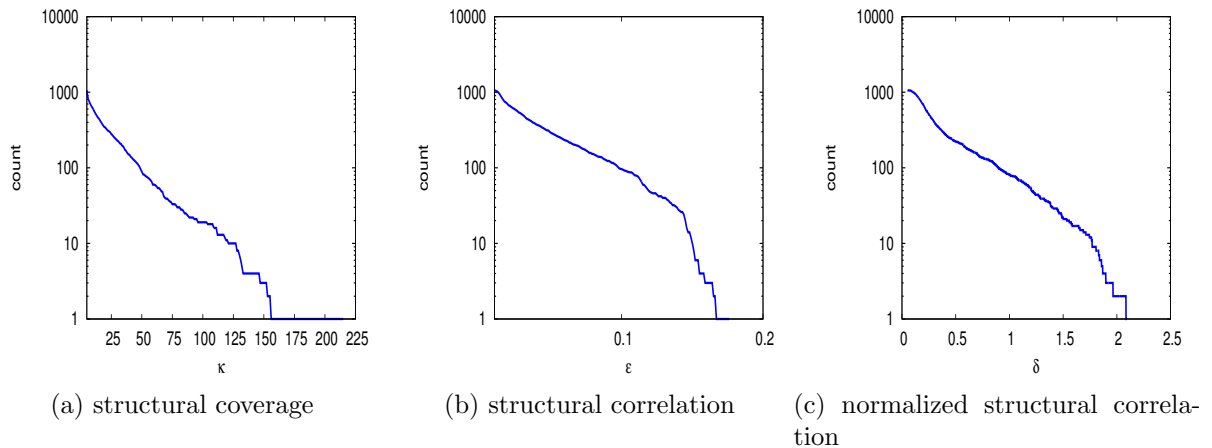


Figure 5.23: Cumulative structural coverage (κ), structural correlation (ϵ), and normalized structural correlation (δ_2)

Table 5.12 shows the correlation between support, structural coverage, structural correlation, and normalized structural correlation in the Human dataset through scatter plots. The support of the attribute sets is significantly correlated with their structural coverage, but not with the structural correlation and the normalized structural correlation. In other words, structural correlation patterns associated to more frequent attribute sets cover more vertices, but it does not lead to a significant correlation between such attribute sets and the formation of dense subgraphs. Moreover, despite the low support of extended attribute sets, of size 4, for example, some of them have high values of structural correlation and normalized structural correlation.

In general, attribute sets with high structural correlation have intermediate or high structural coverage. However, only intermediate values of structural coverage

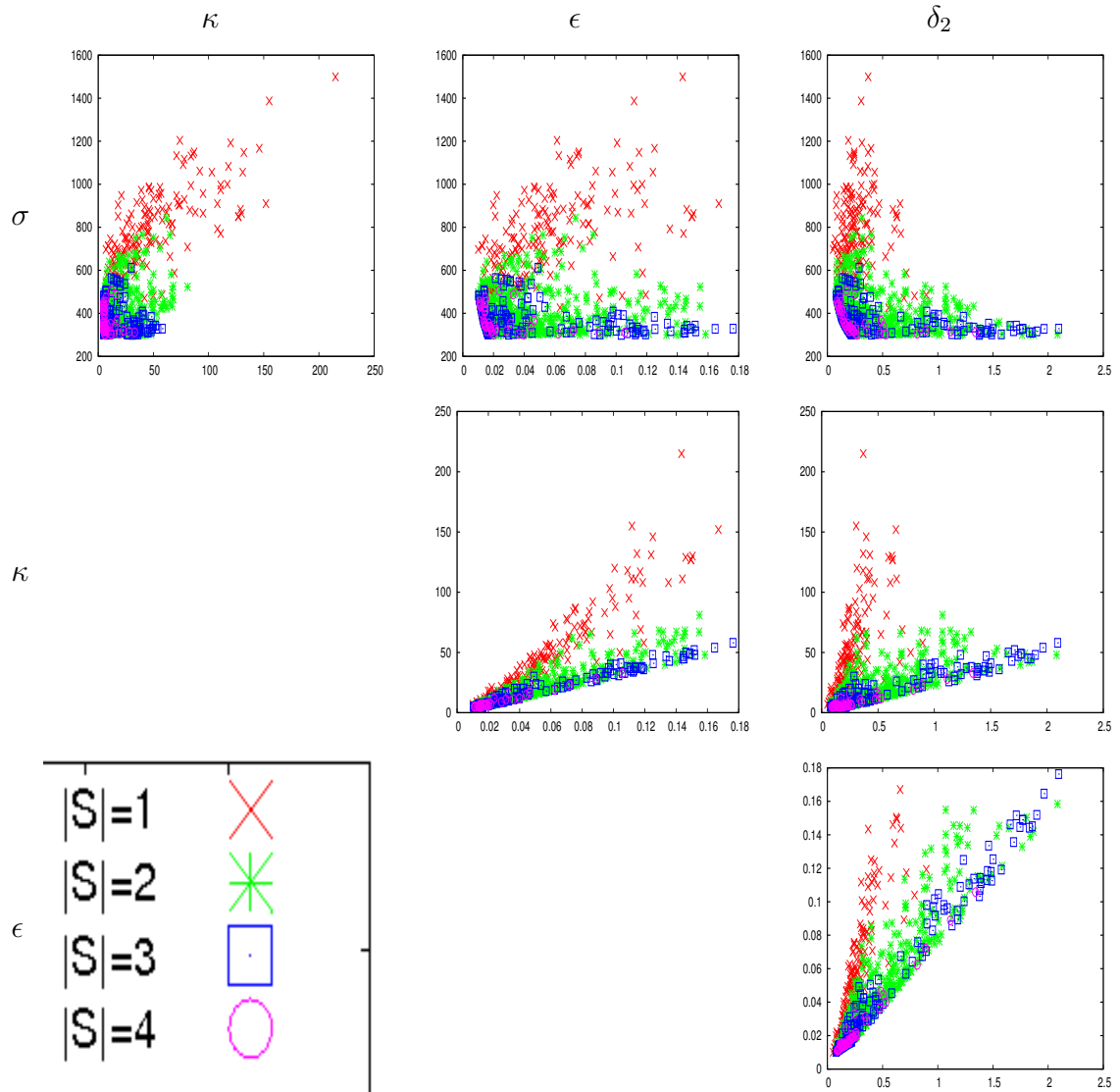


Table 5.12: Scatter plots of the correlations among support (σ), structural coverage (κ), structural correlation (ϵ), and normalized structural correlation for different attribute set sizes ($|S|$) in the Human dataset

are associated to a high normalized structural correlation, what is a direct result of the impact of the support over the structural coverage and the structural correlation function. While the correlation between frequent attribute sets and the formation of dense subgraphs may be biased (see Section 3.2), some low support attribute sets have a significant structural correlation if we consider their intermediate or low support. Moreover, attribute sets with high normalized structural correlation have also high structural correlation, but the opposite is not necessarily true.

5.2 Parameter Sensitivity and Setting

We now assess how different input parameters affect the output of structural correlation pattern mining. Our objective is to provide guidelines for setting the parameters of SCPM. Figures 5.24 and 5.25 show the average structural correlation and normalized structural correlation, respectively, of the complete output (global) and the top-10% attribute sets from the SmallDBLP dataset varying the γ_{min} , min_size , and σ_{min} parameters. SmallDBLP is a smaller version of the DBLP dataset with 32,908 vertices, 82,376 edges, and 11,192 attributes. Default values for γ_{min} , min_size , and σ_{min} are 0.5, 10 and 100. The results show that more restrictive quasi-clique parameters (i.e., high values of γ_{min} and min_size) reduce the average ϵ but may increase δ , since dense subgraphs become less expected. Moreover, high values of σ_{min} are related to high values of structural correlation ϵ . However, such attribute sets also present high values of ϵ_{exp} , leading to low values of normalized structural correlation δ .

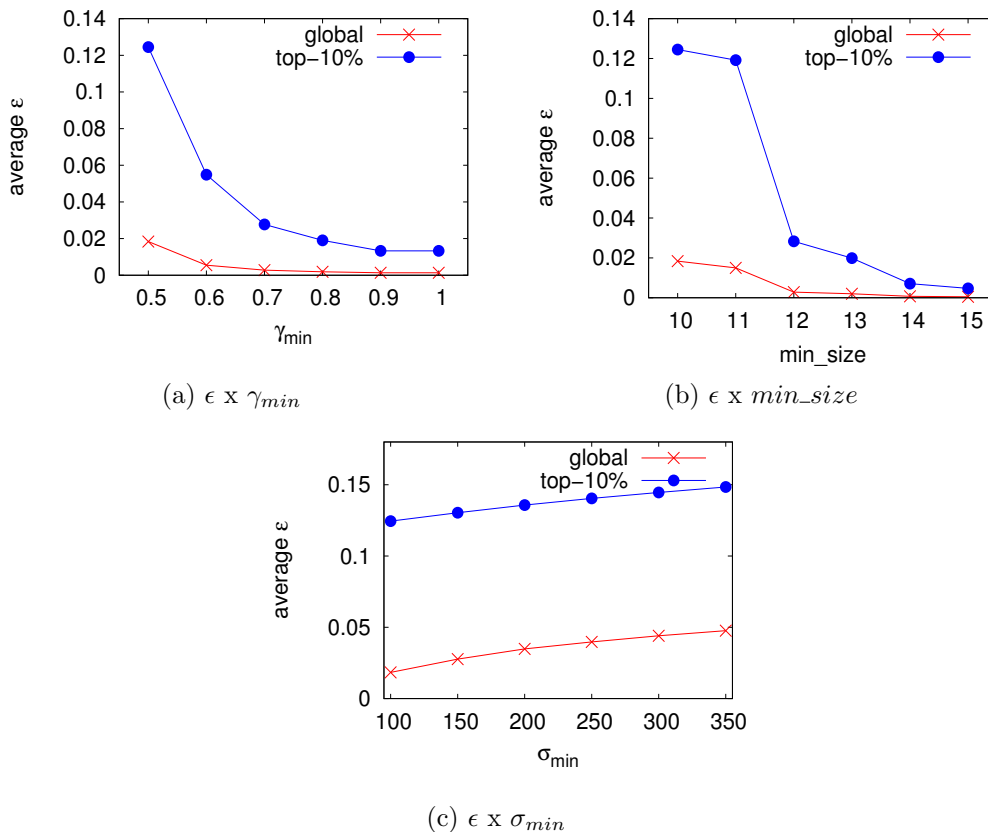


Figure 5.24: Parameter sensitivity w.r.t the minimum structural correlation (ϵ_{min})

SCPM is an exploratory pattern mining method, and thus reasonable values for the different parameters can be obtained by searching the parameter space. The min-

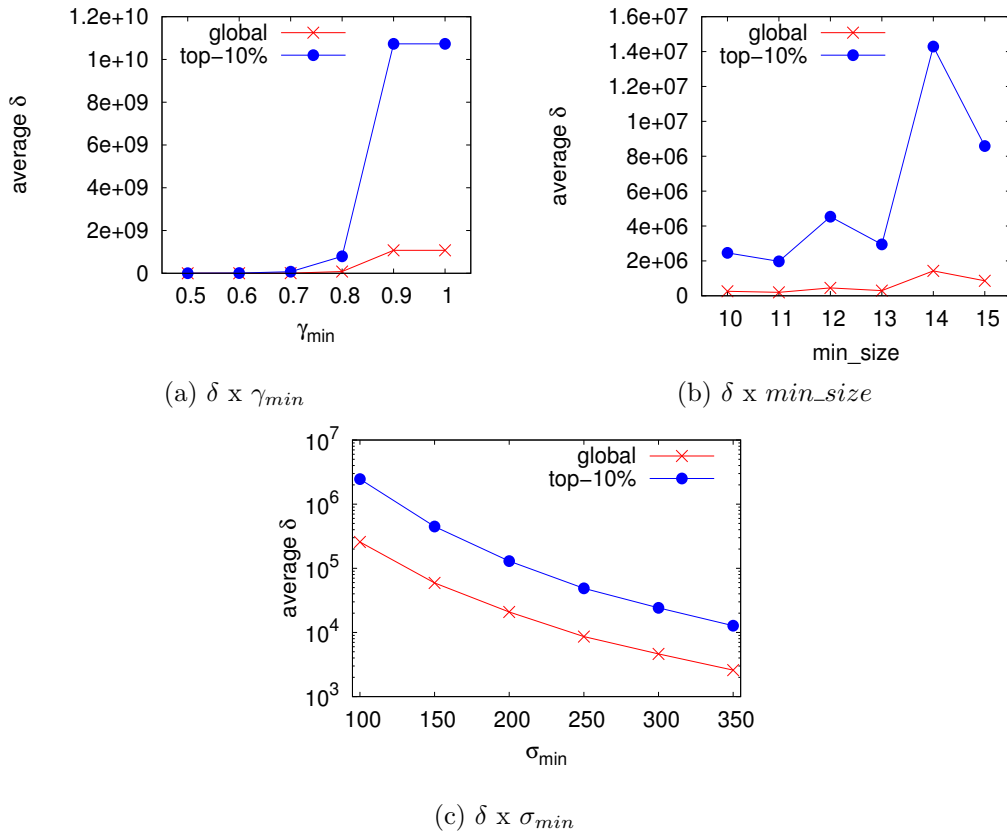


Figure 5.25: Parameter sensitivity w.r.t the minimum normalized structural correlation (δ_{min})

imum density parameter, γ_{min} , and the minimum quasi-clique size, min_size , will depend on the application. For σ_{min} , a useful guideline is to select values that produce a significant expected structural correlation. Infrequent attribute sets may not be expected to induce any dense subgraph. The other parameters (ϵ_{min} , δ_{min} , and k) have as objectives to speedup the algorithm and must be set according to the available computational resources and time.

5.3 Performance Evaluation

In the last sections, we performed case studies in order to show the applicability of the structural correlation pattern mining in real-life scenarios. This section assesses the structural correlation pattern mining from a performance perspective. We evaluate the algorithms proposed in Chapter 4 using real datasets and varying the input parameters.

As discussed in Section 3.4, both the structural correlation pattern mining and the normalized structural correlation pattern mining problems are NP-hard. Therefore,

such tasks require efficient algorithms to enable the processing of large real datasets. Along Chapter 4, we have proposed several techniques in order to improve the performance of the algorithms for the structural correlation pattern mining. In this section, we evaluate such techniques in terms of their execution time.

The datasets used in the performance evaluation of the proposed algorithms are a smaller version of the DBLP dataset (SmallDBLP), and a bigger version (DBLP), which has 108,030 vertices, 276,658 edges, and 23,285 attributes. All experiments presented in this section were executed on a 16-core Intel Xeon 2.4 Ghz with 50GB of RAM. Each result is an average of five executions of a given algorithm using the respective input parameters.

5.3.1 Computing the Structural Correlation

In Section 4.1, we have presented a naive algorithm for structural correlation pattern mining. Algorithm 4 computes the structural correlation of attribute sets by enumerating the complete set of dense subgraphs from the graph induced by a given attribute. However, in Section 4.2, we proposed the two search strategies for computing the structural correlation, which is a subproblem of the standard and normalized structural correlation pattern mining problems. This section evaluates the performance of three algorithms for such task using the SmallDBLP dataset.

We compare the **SCPM-DFS**, the **SCPM-BFS**, and the naive algorithm for structural correlation pattern mining, which we call **Naive**. The **SCPM-DFS** algorithm is a version of the SCPM algorithm using a DFS strategy. Similarly, the **SCPM-BFS** algorithm is a version of SCPM using a BFS strategy. The SCPM algorithm is described in Section 4.2. We vary each parameter of the algorithms keeping the others constant. Default values for the minimum size threshold γ_{min} , the minimum size threshold min_size , and the minimum support threshold σ_{min} are set to 0.5, 11, and 100. Moreover, the minimum structural correlation threshold ϵ_{min} , the minimum normalized structural correlation threshold δ_{min} , and the number of top structural correlation patterns k to be generated are set to 0.1, 1, and 5, respectively, unless stated otherwise.

Figure 5.26 shows the runtime of the **SCPM-DFS**, the **SCPM-BFS**, and the naive algorithm for structural correlation pattern mining (**Naive**) for different values of γ_{min} . The values of min_size , σ_{min} , ϵ_{min} , δ_{min} , and k were set to 11, 100, 0.1, 1, and 5, respectively. We can notice that the **SCPM-DFS** algorithm outperforms the **SCPM-BFS** and the **Naive** algorithm for small values of γ_{min} . Moreover, the **SCPM-BFS** algorithm performs better than the **Naive** algorithm.

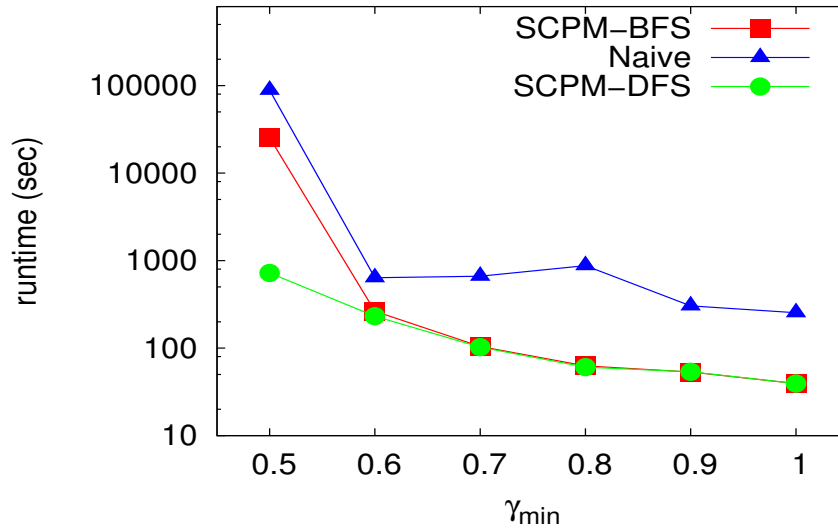


Figure 5.26: Runtime of **SCPM-BFS**, **Naive** and **SCPM-DFS** w.r.t. γ_{min}

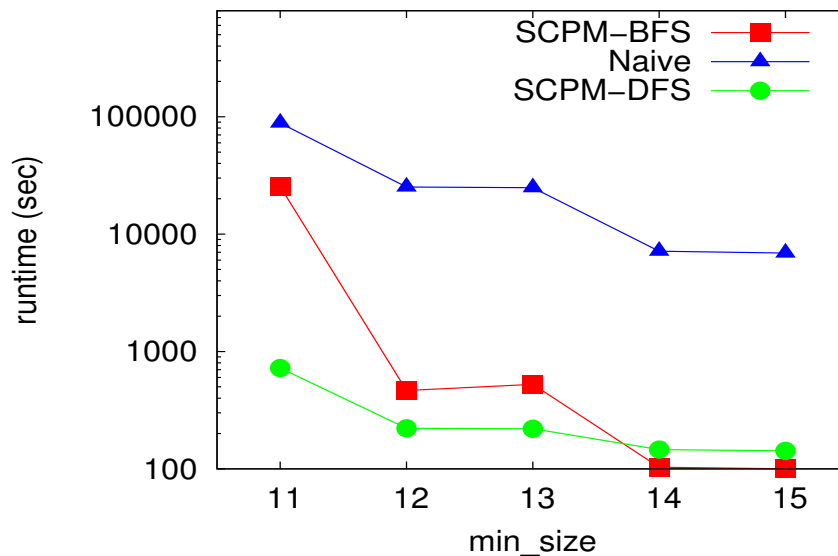


Figure 5.27: Runtime of **SCPM-BFS**, **Naive** and **SCPM-DFS** w.r.t. min_size

Figure 5.27 compares the runtime of the **SCPM-DFS**, the **SCPM-BFS**, and the **Naive** algorithm varying the minimum size threshold min_size . The parameters γ_{min} , σ_{min} , ϵ_{min} , δ_{min} , and k were set to 0.5, 100, 0.1, 1, and 5, respectively. In general, the **SCPM-DFS** algorithm achieves the best results, being more efficient than the **SCPM-BFS** and the **Naive** algorithm. The **SCPM-BFS** algorithm is, again, significantly faster than the **Naive** algorithm.

In Figure 5.28, we present the runtime of the **SCPM-DFS**, the **SCPM-BFS** and

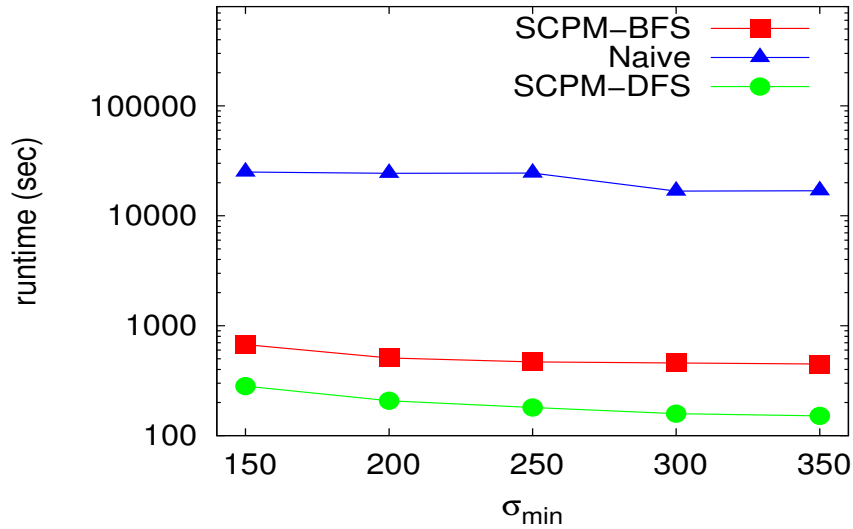


Figure 5.28: Runtime of **SCPM-BFS**, **Naive** and **SCPM-DFS** w.r.t. σ_{min}

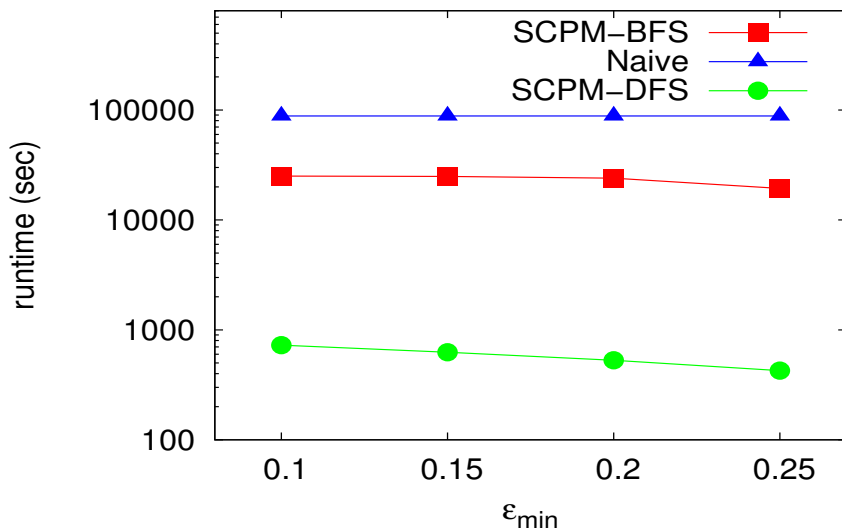


Figure 5.29: Runtime of **SCPM-BFS**, **Naive** and **SCPM-DFS** w.r.t. ϵ_{min}

the **Naive** algorithm for structural correlation pattern mining varying the minimum support threshold σ_{min} . The values of γ_{min} , min_size , ϵ_{min} , δ_{min} and k were set to 0.5, 11, 0.1, 1, and 5, respectively. The **SCPM-DFS** algorithm is the most efficient for all values of σ_{min} .

We also compare the **SCPM-DFS**, the **SCPM-BFS** and the **Naive** algorithm for computing the structural correlation w.r.t. the minimum structural correlation threshold ϵ_{min} . The **SCPM-BFS** and the **SCPM-DFS** algorithms apply a pruning

technique for attribute sets based on the upper bound on structural correlation (see Theorem 7). Therefore, it is interesting to see how such algorithms make use of the ϵ_{min} parameter in order to improve their performances. Figure 5.29 shows the running time of the algorithms for different values of ϵ_{min} . The γ_{min} , min_size , σ_{min} , δ_{min} , and k parameters were set to 0.5, 11, 100, 1 and 5, respectively. The **SCPM-DFS** algorithm provided the best results, followed by the **SCPM-BFS**.

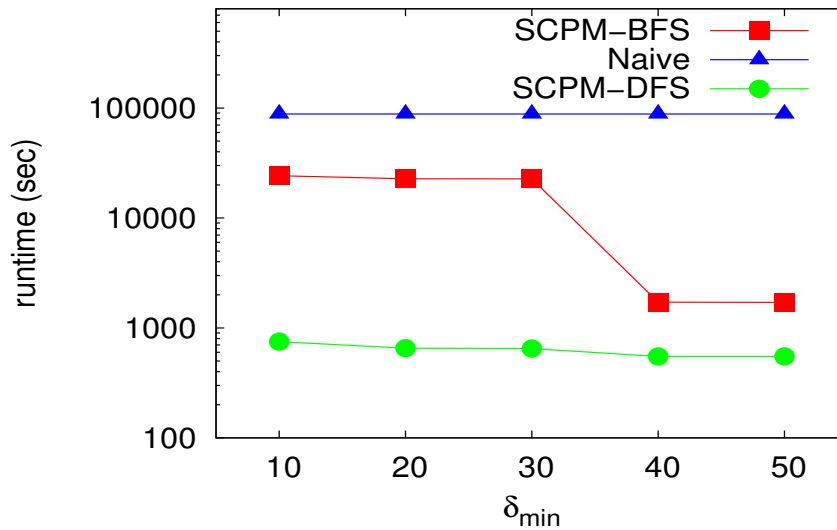


Figure 5.30: Runtime of **SCPM-BFS**, **Naive** and **SCPM-DFS** w.r.t. δ_{min}

Similarly to ϵ_{min} , attribute sets may also be pruned based on a minimum normalized structural correlation threshold (see Theorem 8). In Figure 5.30, we compare the **Naive**, the **SCPM-BFS** and the **SCPM-DFS** algorithm varying the value of δ_{min} . The γ_{min} , min_size , σ_{min} , ϵ_{min} , and k parameters were set to 0.5, 11, 100, 0.1 and 5, respectively. In general, the **SCPM-DFS** algorithm presents the best results. For low values of δ_{min} , the **SCPM-BFS** algorithm is outperformed by the **Naive** algorithm.

Based on the results obtained in this section, the **SCPM-DFS** algorithm (i.e., the SCPM algorithm using a DFS strategy) is the most efficient algorithm for computing the structural correlation. Moreover, both the **SCPM-BFS** and the **SCPM-DFS** algorithms may take advantage of the ϵ_{min} and the δ_{min} parameters in order to execute faster, due to the pruning techniques proposed in Section 4.3.

5.3.2 Sampling

In the last section, we evaluated three algorithms for computing the structural correlation in terms of execution time varying several input parameters. This section

studies the sampling strategies for computing the structural correlation proposed in Section 4.4. Such sampling techniques may enable the computation of the structural correlation of attribute sets using a limited sample of vertices from their respective induced graphs. The deviations of the sampled results from those obtained considering the entire population are estimated by their margin of error. The dataset used in this evaluation is the SmallDBLP dataset.

The sampling techniques are evaluated in terms of the execution time of the algorithms and the average error of their estimates for the structural correlation of attribute sets. More specifically, we are interested in how the maximum estimated error accepted (θ_{max}) affects the performance and the accuracy of the proposed algorithms. In Section 4.4, we described two sampling techniques for computing the structural correlation, one based on a DFS and other based on a BFS strategy for checking whether a vertex is a member of a dense subgraph. We call **SCPM-SAMP-DFS** the version of the SCPM algorithm that computes the structural correlation using sampling and DFS. The **SCPM-SAMP-BFS** algorithm computes the structural correlation using sampling and BFS. The values of γ_{min} , min_size , σ_{min} , ϵ_{min} , δ_{min} , and k are set to 0.5, 11, 100, 0.1, 1, and 5 respectively. Every result is an average of five executions.

Figure 5.31 compares the execution time of the **SCPM-DFS**, the **SCPM-SAMP-BFS**, and the **SCPM-SAMP-DFS** algorithm for different values of θ_{max} . **SCPM-DFS** computes the exact structural correlation and their execution time does not depend on θ_{max} . We can notice that the use of sampling implies in a significant reduction of the execution time of the algorithms when compared to **SCPM-DFS**. Moreover, **SCPM-DFS-SAMP** outperforms **SCPM-BFS**.

It is also relevant to consider the accuracy of the estimates produced by the sampling techniques. Figure 5.32 shows the mean squared error of the estimates of the structural correlation using sampling. Since both **SCPM-SAMP-BFS** and **SCPM-SAMP-DFS** presented similar results in terms of average error, we decided to omit the results for **SCPM-SAMP-BFS**. The use of sampling for computing the structural correlation produces accurate results. As expected, the θ_{max} parameter regulates the average error of the estimates provided.

The results discussed in this section show that estimating the structural correlation of attribute sets through sampling is a good alternative, both in terms of execution time and accuracy. The estimates obtained have a low average error and can be computed efficiently. In the next section, we evaluate the algorithm for the discovery of the top-k structural correlation patterns, proposed in Section 4.5.

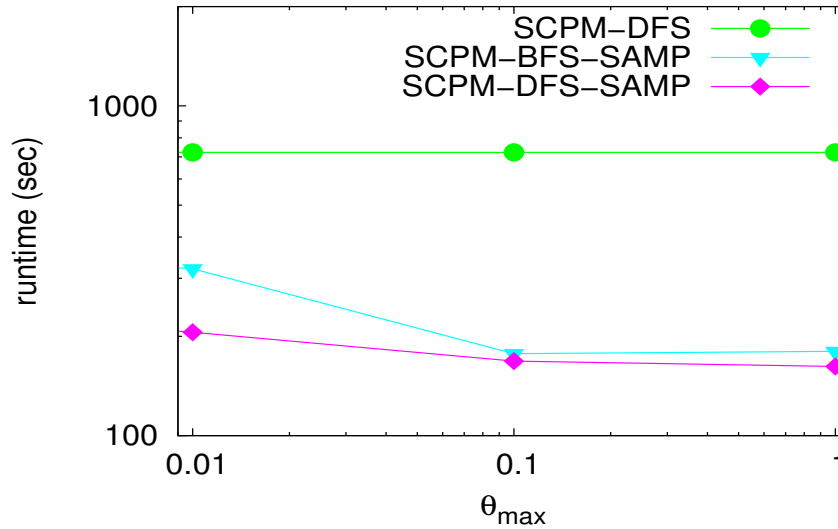


Figure 5.31: Runtime of **SCPM-DFS**, **SCPM-BFS-SAMP** and **SCPM-DFS-SAMP** w.r.t. θ_{max}

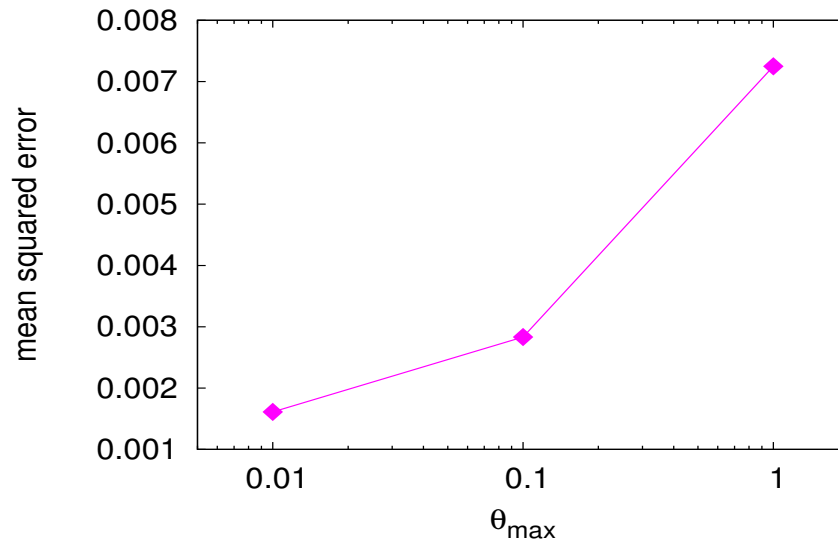


Figure 5.32: Mean squared error of **SCPM-SAMP-DFS** w.r.t. θ_{max}

5.3.3 Discovering the Top-K Structural Correlation Patterns

In Section 4.5, we proposed the enumeration of a limited set of most interesting structural correlation patterns as an alternative to the generation of the complete set of patterns. The main motivation for this approach is the high computation cost of discovering quasi-cliques (see Section 2.2 for more details). This section evaluates our algorithm for identifying the top structural correlation patterns in terms of size and

density.

Figure 5.33 shows the execution time of the **Naive** and the **SCPM-DFS** algorithms varying the number of top structural correlation patterns to identified (k). Since the **Naive** algorithm always computes the complete set of structural correlation patterns, it is not affected by the value of k . The results for the **SCPM-BFS** algorithm are not presented because the focus of this section is on the evaluation of the technique for the discovery of the top-k patterns, which is applied by both the **SCPM-DFS** and the **SCPM-BFS**. The parameters γ_{min} , min_size , σ_{min} , ϵ_{min} , δ_{min} and k are set to 0.5, 11, 100, 0.1, and 1, respectively. The inset also shows the execution time of **SCPM-DFS** using a linear scale for the y-axis.

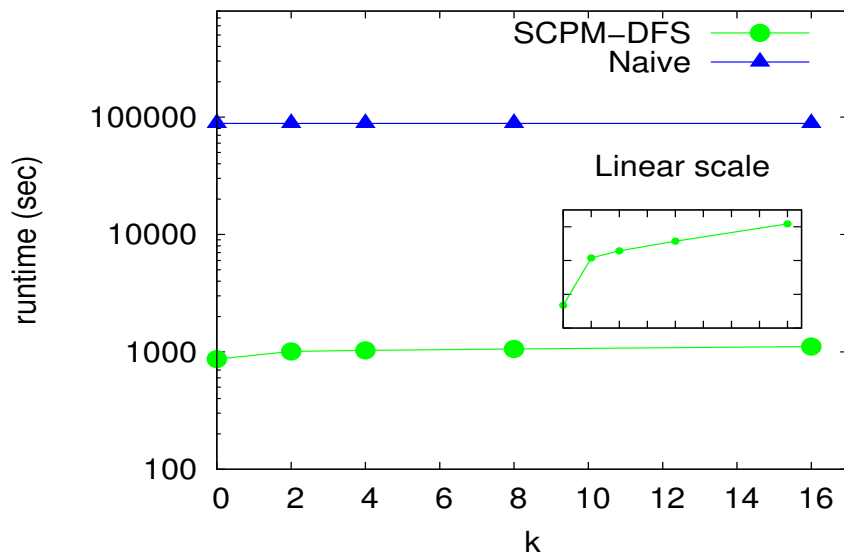


Figure 5.33: Runtime of the **SCPM-DFS** and the **Naive** algorithm w.r.t. k

It is interesting to notice that the running time of the **SCPM-DFS** algorithm presents a decreasing cost when k is increased. In other words, the highest is the value of k , the lower is the cost of increasing it by one unit. As expected, by setting low values of k , the execution time of the algorithm can be reduced significantly.

In the next section, we evaluate the parallel algorithms for structural correlation pattern mining presented in Section 4.7. We study the execution time and the scalability of such algorithms varying the number of processors available.

5.3.4 Parallel Algorithms

In Section 4.7 we proposed parallel algorithms for computing the structural correlation of attribute sets and for identifying the top-k structural correlation patterns in terms

of size and density. The main motivation for the design of parallel algorithms for structural correlation pattern mining is the high computational cost associated to such tasks, as discussed in Chapter 3. This section evaluates how the proposed algorithms perform in terms of execution time and scalability using the SmallDBLP dataset. The evaluations consider only the part of the respective algorithm that has been parallelized.

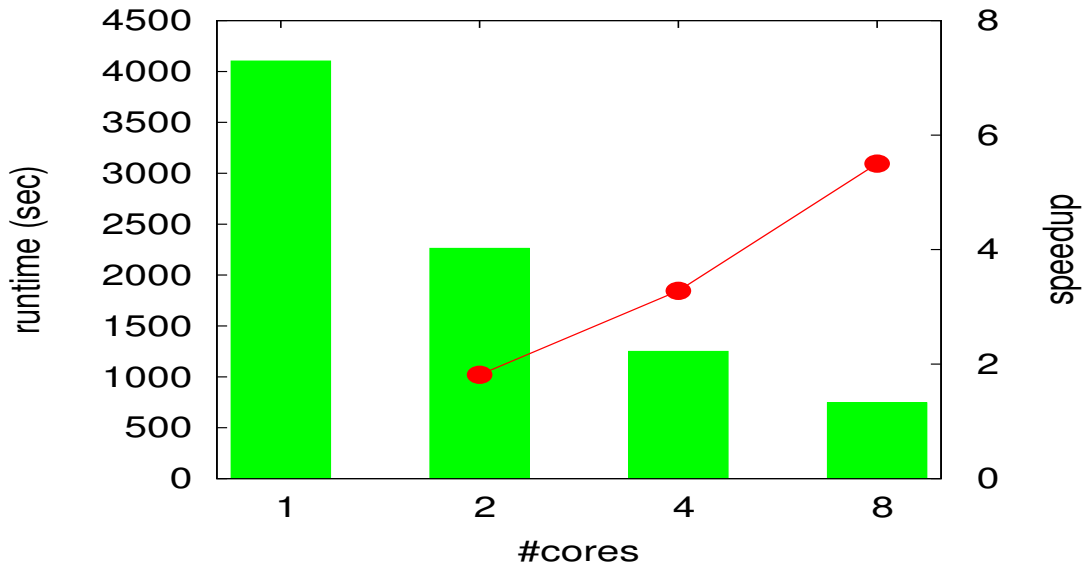


Figure 5.34: Runtime and speedup of **PAR-SCPM-DFS** w.r.t. the number of cores

Figure 5.34 shows the runtime and the scalability of the parallel DFS algorithm for structural correlation pattern mining (**PAR-SCPM-DFS**) for different numbers of cores. The parameters γ_{min} , min_size , σ_{min} , ϵ_{min} , δ_{min} and k were set to 5, 0.5, 400, 0.1, 1, and 5, respectively. We can notice that the algorithm scales well while the number of cores increases. **PAR-SCPM-DFS** is up to 6 times faster than **SCPM-DFS** when 8 cores are available.

In Figure 5.35, we evaluate the runtime and the scalability of the parallel BFS algorithm for structural correlation pattern mining (**PAR-SCPM-BFS**) varying the number of cores. The values of γ_{min} , min_size , σ_{min} , ϵ_{min} , δ_{min} , and k were set to 7, 0.5, 400, 0.1, 1, and 5, respectively. **PAR-SCPM-BFS** presents a super linear speedup (i.e., it is more than p times faster its sequential version when p cores are available). When using two cores, the algorithm is about 80 times faster than its sequential version. Nevertheless, such results are due to the poor performance of the **SCPM-BFS** algorithm, as discussed in Section 5.3.1.

In Figure 5.36, we present the runtime and the speedup of **PAR-SAMP-SCPM-BFS**, which is a parallel sampling algorithm for structural correlation pattern mining, varying the number of cores available. The values of γ_{min} , min_size , σ_{min} , ϵ_{min} , and

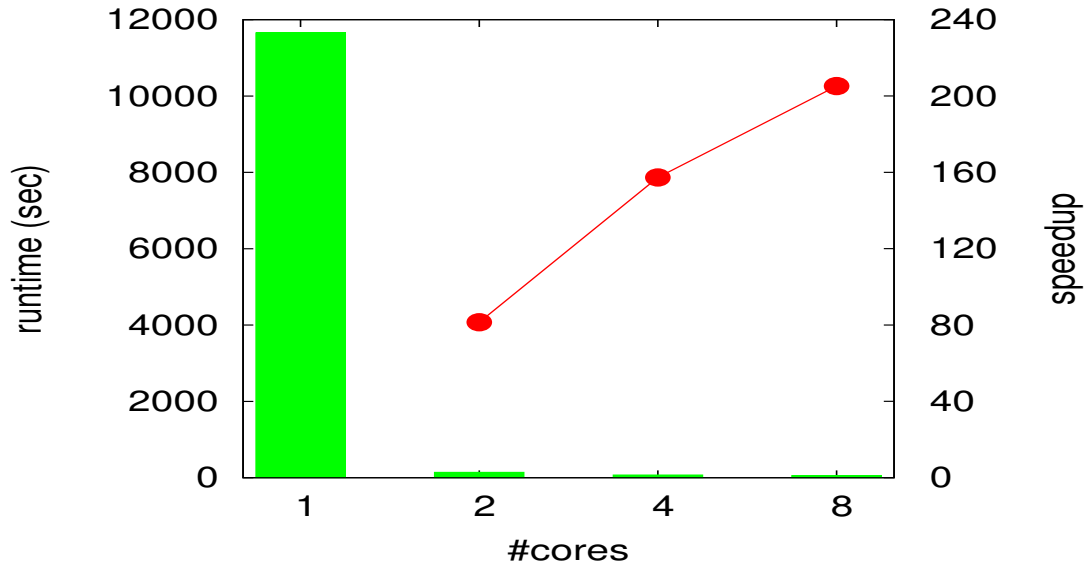


Figure 5.35: Runtime and speedup of **PAR-SCPM-BFS** w.r.t. the number of cores

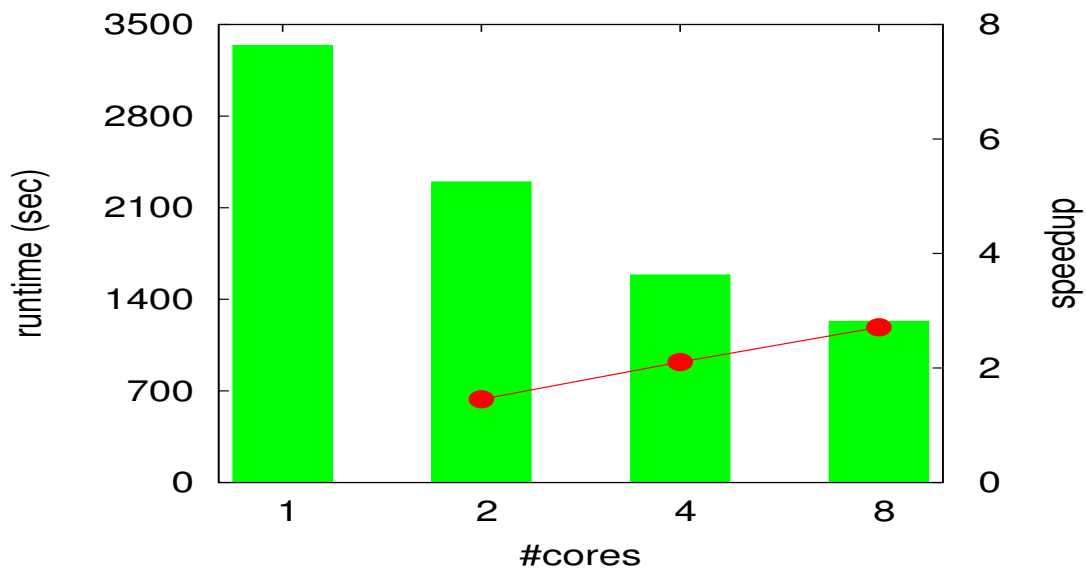


Figure 5.36: Runtime and speedup of the **PAR-SAMP-SCPM-BFS** w.r.t. the number of cores

θ_{max} were set to 3, 0.5, 400, 0, and 0.01 respectively. **PAR-SAMP-SCPM-BFS** scalability is worse than **PAR-SCPM-BFS** and **PAR-SCPM-DFS**. When 8 threads are available, **PAR-SAMP-SCPM-BFS** is only 2.71 times faster than its sequential version. This result may be due to insufficient parallelism [Kumar et al., 1994]. Similar results were found for **PAR-SAMP-SCPM-DFS**, as shown in Figure 5.37. In general, the benefits of parallelization to sampling algorithms are small.

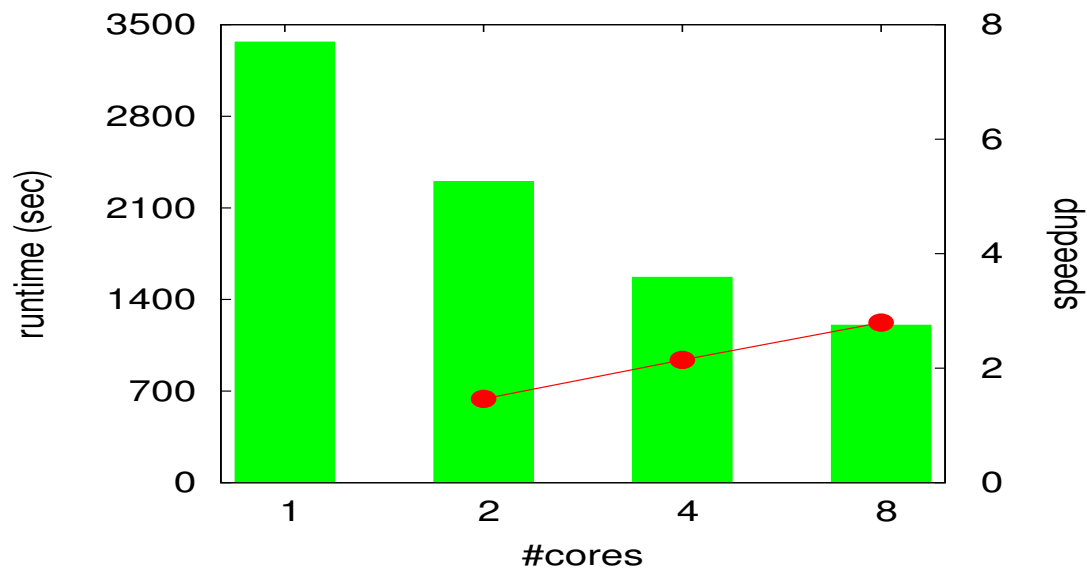


Figure 5.37: Runtime and speedup of the **PAR-SAMP-SCPM-DFS** w.r.t. the number of cores

This section evaluated the performance of parallel algorithms for structural correlation pattern mining using real datasets. We have shown that all the algorithms can take advantage of multiple processors in order to improve their execution times. However, the combination of sampling and parallelization in structural correlation pattern mining has not achieved good scalability in general.

5.3.5 Discussion

This chapter studied the structural correlation pattern mining in terms of the applicability of the proposed patterns and the performance of the presented algorithms. Real datasets were applied in order to characterize the knowledge provided by the structural correlation pattern mining. Moreover, we have evaluated the pruning, sampling, and parallelization strategies discussed in Chapter 4. The results show that such strategies may enable the analysis of large datasets efficiently.

The next chapter concludes this work summarizing its main contributions and limitations. Moreover, some directions for future research are proposed.

Chapter 6

Conclusions

In this work, we studied the problem of correlating vertex attributes and dense subgraphs in large attributed graphs. Vertex attributes play an important role in several real graphs. Moreover, many of these graphs are known to be organized into dense subgraphs, which are sets of vertices strongly connected among themselves. Therefore, attributed graphs may enable the discovery of relevant knowledge regarding the relationships between vertex attributes and dense subgraphs. This chapter summarizes the main contributions, limitations, and future research directions of this thesis.

6.1 Summary of Contributions

This thesis describes the study of the correlation between attribute sets and the formation of dense subgraphs, which we call structural correlation pattern mining. The main contributions of this work can be summarized as follows:

- **General problem proposal:** We propose the general problem of correlating vertex attributes and the existence of dense subgraphs in attributed graphs. As far as we know, there is no previous work on this problem in the literature. The objective of the structural correlation is to measure how a set of attributes induces dense subgraphs in large attributed graphs. A structural correlation pattern is a dense subgraph induced by a particular attribute set. Therefore, the structural correlation pattern mining provides knowledge at both the level of the attribute sets and the dense subgraphs induced by them.
- **Modeling:** Based on the problem proposal, we defined a structural correlation function and a structural correlation pattern combining two existing data mining

patterns (frequent itemsets and quasi-cliques). Moreover, we proposed normalization approaches in order to assess how the structural correlation of a given attribute set deviates from the expected. The structural correlation pattern mining is shown to be NP-hard, requiring efficient and scalable algorithm for its application to large datasets.

- **Algorithm design:** We designed algorithms for structural correlation pattern mining. The algorithms explore different search, pruning, sampling, and parallelization strategies to compute the structural correlation efficiently. Moreover, we proposed the identification of the top structural correlation patterns, instead of the complete set of structural correlation patterns, in order to enable the processing of large graphs in a feasible time.
- **Applications and evaluation:** We applied the structural correlation pattern mining to several real datasets from different domains. Using such datasets, we evaluated the performance of the proposed algorithms. We also conducted case studies in order to show the applicability of the structural correlation pattern mining in real-life scenarios. We found attribute sets with structural correlation much higher than expected in all datasets studied. Such attribute sets carry a relevant knowledge about the relation between attributes and dense subgraphs. Moreover, the structural correlation patterns discovered provide interesting knowledge regarding the dense subgraphs induced by attribute sets.

6.2 Limitations

At this point, it is important to highlight the main limitations of this work. We address some of these limitations as future work in the next section. We summarize the limitations of this work as follows:

- **Performance:** Along this thesis, we argued that the proposed algorithms enable the analysis of large attributed graphs. In our case studies, we applied the proposed algorithms to attributed graphs with hundreds of thousands of vertices, edges, and attributes. However, we know that the proposed algorithms may not be able to process some real datasets. We tried to analyze an attributed graph generated from the Wikipedia, with 3,135,812 vertices, 54,877,914 edges, and 1,756,599 attributes, for example, without success.
- **Applications:** In this work, we applied the structural correlation pattern mining to several real datasets from different domains. The case studies presented

provided qualitative insights regarding the applicability of the structural correlation pattern mining to real-life attributed graphs. On the other hand, we know that it would be relevant to verify the applicability of the structural correlation pattern mining quantitatively. In particular, the use of the structural correlation and structural correlation patterns to new and existing problems is of special interest.

- **Comparison with other methods:** The proposal of the structural correlation pattern mining is one of the contributions of this work. In an extensive search for similar methods in the literature, we could not find any equivalent problem. Related problems include the graph clustering, specially the versions of the problem that consider not only the graph topology but also the vertex attributes. In this work, we did not compare the proposed patterns with any existing data mining pattern from the literature.

6.3 Future Work

In the development of this work, we identified several directions for future research. We summarize them as follows:

- **Pruning:** One of the contributions of this work is the proposal of pruning techniques for the structural correlation pattern mining. Nevertheless, we could find other pruning opportunities that were not explored in this work. In particular, the fact that the structural correlation patterns induced by an attribute set S_i are subsets of the structural correlation patterns induced by an attribute set S_j , such that $S_j \subseteq S_i$, can be used as basis for new pruning techniques. Moreover, it would be relevant to mine closed and maximal structural correlation patterns as means to reduce the number of patterns generated as output.
- **Distributed algorithms:** As mentioned in the last section, we could find a very large dataset that could not be processed by the proposed algorithms for structural correlation pattern mining. Such dataset motivates the design of distributed algorithms for structural correlation pattern mining. Distributed algorithms may handle datasets even larger than the ones used in this work.
- **Diversity measures for structural correlation patterns:** In this work, we proposed mining the top-k largest and densest structural correlation patterns in order to reduce both execution time and output volume. However, we noticed

that some top patterns are very similar, which is a motivation to the application of diversity measures in the identification of the most interesting structural correlation patterns.

- **More complex attribute sets:** Attribute sets are considered as a *bag-of-words* in structural correlation pattern mining. Nevertheless, more complex models may enhance the description power of structural correlation patterns. In particular, attribute sets may be n-grams or multivalued attributes. Moreover, structural correlation patterns may be mined considering multiple relations, as in *multi-relational data mining* [Dzeroski, 2003; Wrobel, 2000].
- **Applications:** As discussed in the last section, one of the main limitations of this work is the absence of quantitative evidences of the utility of the structural correlation pattern mining. Nevertheless, we identified several existing problems that can be solved using the knowledge provided by the structural correlation pattern mining, such as:
 - **Graph clustering:** Structural correlation patterns can be applied in the identification of graph clusters regarding both the graph topology and vertex attributes. In particular, we believe that such patterns can be percolated (i.e., merged recursively) in the generation of clusters that have high cohesion and are composed by vertices that share attribute sets significantly correlated with the formation of dense subgraphs.
 - **Relational learning:** The knowledge provided by the structural correlation pattern has promising applications in relational learning problems. In link prediction, for example, attribute sets can be weighted according to their structural correlation in order to predict the connections between vertices considering both the graph structure and vertex attributes that induce dense subgraphs. Moreover, in vertex classification problems, vertices may be assigned to one or more classes based not only on the classes of their neighbors, but also on how such classes are related to the formation of dense subgraphs.
 - **Graph summarization and visualization:** Structural correlation patterns may be considered as new entities that represent their respective vertices. Since each structural correlation pattern has an attribute set associated with it, such entities may be labelled based on the attributes that induced them. Therefore, a new graph in which vertices are structural correlation patterns, edges are connections among vertices from different pat-

terns, and vertex attributes are the attribute sets that induced the respective pattern, may be used as a summarized view of the original attributed graph. Moreover, such graph may be visualized, enabling a better understanding of the relationships between attributes and dense subgraphs.

Bibliography

- Abello, J., Resende, M. G. C., and Sudarsky, S. (2002). Massive quasi-clique detection. In *Proceedings of the Latin American Symposium on Theoretical Informatics*, pages 598--612.
- Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the International Conference on Management of Data*, pages 207--216.
- Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the International Conference on Very Large Databases*, pages 487--499.
- Alba, R. (1973). A graph-theoretic definition of a sociometric clique. *The Journal of Mathematical Sociology*, 3(1):113--126.
- Albert, R. (2005). Scale-free networks in cell biology. *Arxiv preprint q-bio/0510054*.
- Albert, R., Jeong, H., and Barabási, A. (1999). Internet: Diameter of the world-wide web. *Nature*, 401(6749):130--131.
- Albert, R., Jeong, H., and Barabási, A. (2000). Error and attack tolerance of complex networks. *Arxiv preprint cond-mat/0008064*.
- Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Taylor & Francis, New York, NY, USA.
- Anagnostopoulos, A., Kumar, R., and Mahdian, M. (2008). Influence and correlation in social networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 7--15.
- Anderson, T. and Finn, J. (1996). *The new statistical analysis of data*. Springer Verlag, New York, NY, USA.

- Beck, G. and Habicht, G. (1996). Immunity and the invertebrates. *Scientific American*, 275(5):60.
- Borgatti Martin, G. and Stephen, P. (1990). LS sets, lambda sets and other cohesive subsets. *Social Networks*, 12(4):337--357.
- Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30:107--117.
- Buehrer, G., Parthasarathy, S., and Chen, Y.-K. (2006). Adaptive parallel graph mining for cmp architectures. In *Proceedings of the International Conference on Data Mining*, pages 97--106.
- Burgisser, P., Clausen, M., and Shokrollahi, M. (1997). *Algebraic complexity theory*. Springer Verlag, New York, NY, USA.
- Carrington, P. J. (2005). *Models and Methods in Social Network Analysis (Structural Analysis in the Social Sciences)*. Cambridge University Press, New York, NY, USA.
- Ceglar, A. and Roddick, J. F. (2006). Association mining. *ACM Comput. Surv.*, 38(2):1--42.
- Chakrabarti, D. (2004). Autopart: parameter-free graph partitioning and outlier detection. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 112--124.
- Chakrabarti, D. and Faloutsos, C. (2006). Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1):2.
- Chan, R., Yang, Q., and Shen, Y.-D. (2003). Mining high utility itemsets. In *Proceedings of the International Conference on Data Mining*, pages 19--22.
- Dourisboure, Y., Geraci, F., and Pellegrini, M. (2009). Extraction and classification of dense implicit communities in the web graph. *ACM Trans. Web*, 3(2):1--36.
- Džeroski, S. (2003). Multi-relational data mining: an introduction. *SIGKDD Explor. Newsl.*, 5(1):1--16.
- Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. (1996). Advances in knowledge discovery and data mining. pages 1--34. American Association for Artificial Intelligence, Menlo Park, CA, USA.

- Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3-5):75--174.
- Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.
- Ge, R., Ester, M., Gao, B. J., Hu, Z., Bhattacharya, B., and Ben-Moshe, B. (2008). Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. *ACM Trans. Knowl. Discov. Data*, 2(2):1--35.
- Gibson, D., Kumar, R., and Tomkins, A. (2005). Discovering large dense subgraphs in massive graphs. In *Proceedings of the International Conference on Very Large Databases*, pages 721--732.
- Girvan, M. and Newman, M. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821--7826.
- Grigoriev, A. (2001). A relationship between gene expression and protein interactions on the proteome scale: analysis of the bacteriophage T7 and the yeast *Saccharomyces cerevisiae*. *Nucleic Acids Research*, 29(17):3513--3519.
- Guan, Z., Wu, J., Zhang, Q., Singh, A., and Yan, X. (2011). Assessing and ranking structural correlations in graphs. In *Proceedings of the International Conference on Management of Data*, pages 937--948.
- Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., and Sharma, R. S. (2003). Discovering all most specific sentences. *ACM Trans. Database Syst.*, 28(2):140--174.
- Gyöngyi, Z. and Garcia-Molina, H. (2005). Link spam alliances. In *Proceedings of the International Conference on Very Large Databases*, pages 517--528.
- Han, J. (2005). *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, USA.
- Hand, D. J., Smyth, P., and Mannila, H. (2001). *Principles of data mining*. MIT Press, Cambridge, MA, USA.
- Hanisch, D., Zien, A., Zimmer, R., and Lengauer, T. (2002). Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(1):145--154.
- Hartwell, L., Hopfield, J., Leibler, S., Murray, A., et al. (1999). From molecular to modular cell biology. *Nature*, 402(6761):47--52.

- Hipp, J., Güntzer, U., and Nakhaeizadeh, G. (2000). Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explor. Newsl.*, 2(1):58–64.
- Hu, H., Yan, X., Huang, Y., Han, J., and Zhou, X. J. (2005). Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21:213–221.
- Hu, Y., Chen, H., Zhang, P., Li, M., Di, Z., and Fan, Y. (2008). Comparative definition of community and corresponding identifying algorithm. *Physical Review E*, 78(2):26121–26127.
- Jeong, H., Mason, S., Barabási, A., and Oltvai, Z. (2001). Lethality and centrality in protein networks. *Nature*, 411(6833):41–42.
- Jiang, D. and Pei, J. (2009). Mining frequent cross-graph quasi-cliques. *ACM Trans. Knowl. Discov. Data*, 2(4):1–42.
- Khan, A., Yan, X., and Wu, K.-L. (2010). Towards proximity pattern mining in large graphs. In *Proceedings of the International Conference on Management of Data*, pages 867–878.
- Kumar, V., Grama, A., Gupta, A., and Karypis, G. (1994). *Introduction to parallel computing: design and analysis of algorithms*. Benjamin-Cummings Publishing, Redwood City, CA, USA.
- Lappas, T., Liu, K., and Terzi, E. (2009). Finding a team of experts in social networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 467–476.
- Leskovec, J. (2008). *Dynamics of large networks*. PhD thesis, Pittsburgh, PA, USA.
- Liu, G. and Wong, L. (2008). Effective pruning techniques for mining quasi-cliques. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases*, pages 33–49.
- Mannila, H. and Toivonen, H. (1997). Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Discov.*, 1(3):241–258.
- McPherson, M., Smith-Lovin, L., and Cook, J. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444.

- Menezes, G. V., Ziviani, N., Laender, A. H., and Almeida, V. (2009). A geographical analysis of knowledge production in computer science. In *Proceedings of the International Conference on World Wide Web*, pages 1041--1050.
- Moser, F., Colak, R., Rafiey, A., and Ester, M. (2009). Mining cohesive patterns from graphs with feature vectors. In *Proceedings of the International Conference on Data Mining*, pages 593--604.
- Moser, F., Ge, R., and Ester, M. (2007). Joint cluster analysis of attribute and relationship data without a priori specification of the number of clusters. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 510--519.
- Mougel, P.-N., Plantevit, M., Rigotti, C., Gandrillon, O., and Boulicaut, J.-F. (2010). Constraint-based mining of sets of cliques sharing vertex properties. In *Proceedings of the Workshop on Analysis of Complex Networks*, pages 48--62.
- Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45(2):167--256.
- Newman, M. E. J. (2004). Detecting community structure in networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):321--330.
- Ng, R. T., Lakshmanan, L. V. S., Han, J., and Pang, A. (1998). Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of the International Conference on Management of Data*, pages 13--24.
- Palla, G., Derényi, I., Farkas, I., and Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814--818.
- Pei, J., Jiang, D., and Zhang, A. (2005). On mining cross-graph quasi-cliques. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 228--238.
- Pons-Porrata, A., Berlanga-Llavori, R., and Ruiz-Shulcloper, J. (2007). Topic discovery based on text mining techniques. *Inf. Process. Manage.*, 43(3):752--768.
- Pôssas, B., Ziviani, N., Meira, Jr., W., and Ribeiro-Neto, B. (2005). Set-based vector model: An efficient approach for correlation-based ranking. *ACM Trans. Inf. Syst.*, 23(4):397--429.

- Puig-Centelles, A., Ripolles, O., and Chover, M. (2008). Surveying the identification of communities. *Int. J. Web Based Communities*, 4(3):334--347.
- Scott, J. P. (2000). *Social Network Analysis: A Handbook*. Sage Publications, London, UK.
- Seidman, S. (1983a). LS sets as cohesive subsets of graphs and hypergraphs. *Mathematical Social Sciences*, 6(1):87--91.
- Seidman, S. (1983b). Network structure and minimum degree. *Social Networks*, 5(3):269--287.
- Seidman, S. and Foster, B. (1978). A graph-theoretic generalization of the clique concept. *The Journal of Mathematical Sociology*, 6(1):139--154.
- Sese, J., Seki, M., and Fukuzaki, M. (2010). Mining networks with shared items. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 1681--1684.
- Shi, X., Leskovec, J., and McFarland, D. A. (2010). Citing for high impact. In *Proceedings of the Joint Conference on Digital Libraries*, pages 49--58.
- Shyamsundar, R., Kim, Y., Higgins, J., Montgomery, K., Jorden, M., Sethuraman, A., Van De Rijn, M., Botstein, D., Brown, P., and Pollack, J. (2005). A DNA microarray survey of gene expression in normal human tissues. *Genome Biology*, 6(3):22--29.
- Silva, A., Meira, Jr., W., and Zaki, M. J. (2010). Structural correlation pattern mining for large graphs. In *Proceedings of the Workshop on Mining and Learning with Graphs*, pages 119--126.
- Silva, A., Meira, Jr., W., and Zaki, M. J. (2012). Mining attribute-structure correlated patterns in large attributed graphs. *PVLDB*, 5(5):466--477.
- Spirin, V. and Mirny, L. A. (2003). Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123--12128.
- Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley Longman Publishing, Boston, MA, USA.
- Thabtah, F. (2007). A review of associative classification mining. *Knowl. Eng. Rev.*, 22(1):37--65.

- Tomita, E., Tanaka, A., and Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.*, 363:28--42.
- Travers, J. and Milgram, S. (1969). An experimental study of the small world problem. *Sociometry*, 32(4):425--443.
- Ulitsky, I. and Shamir, R. (2007). Identification of functional modules using network topology and high-throughput data. *BMC Systems Biology*, 1(8):1--17.
- Valiant, L. (1979). The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189--201.
- Wilkinson, B. and Allen, M. (2004). *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers (2nd Edition)*. Prentice-Hall, Upper Saddle River, NJ, USA.
- Witten, I. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA, USA.
- Wonnacott, T. and Wonnacott, R. (1985). *Introductory statistics*. Wiley, New York, NY, USA.
- Wrobel, S. (2000). Relational data mining. pages 74--99. Springer-Verlag New York, Inc., New York, NY, USA.
- Wuchty, S., Oltvai, Z., and Barabási, A. (2003). Evolutionary conservation of motif constituents in the yeast protein interaction network. *Nature Genetics*, 35(2):176--179.
- Xu, X., Yuruk, N., Feng, Z., and Schweiger, T. A. J. (2007). Scan: a structural clustering algorithm for networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 824--833.
- Yang, G. (2004). The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 344--353.
- Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Trans. on Knowl. and Data Eng.*, 12:372--390.

- Zaki, M. J., Parthasarathy, S., Li, W., and Ogihara, M. (1997). Evaluation of sampling for data mining of association rules. In *Proceedings of the International Workshop on Research Issues in Data Engineering*, pages 42--51.
- Zeng, Z., Wang, J., Zhou, L., and Karypis, G. (2006). Coherent closed quasi-clique discovery from large dense graph databases. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 797--802.
- Zhang, W., Yoshida, T., Tang, X., and Wang, Q. (2010). Text clustering using frequent itemsets. *Know.-Based Syst.*, 23(5):379--388.
- Zhao, Y., Zhang, C., and Zhang, S. (2006). Efficient frequent itemsets mining by sampling. In *Proceedings of the Conference on Advances in Intelligent IT*, pages 112--117.
- Zhou, Y., Cheng, H., and Yu, J. X. (2009). Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718--729.