



# Topological and Domain Knowledge-based Subgraph Mining: Application on Protein 3D-Structures

Wajdi Dhifli

## ► To cite this version:

Wajdi Dhifli. Topological and Domain Knowledge-based Subgraph Mining: Application on Protein 3D-Structures. Machine Learning [cs.LG]. Université Blaise Pascal - Clermont-Ferrand II, 2013. English. NNT : . tel-00922209

**HAL Id: tel-00922209**

**<https://theses.hal.science/tel-00922209>**

Submitted on 24 Dec 2013

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITY OF CLERMONT-FERRAND II

LIMOS - CNRS UMR 6158

Laboratoire d'Informatique, de Modélisation et d'Optimisation des  
Systèmes

# P H D T H E S I S

To obtain the title of

Doctor of Philosophy

Specialty : COMPUTER SCIENCE

Defended by

Wajdi DHIFLI

---

## Topological and Domain Knowledge-based Subgraph Mining: Application on Protein 3D-Structures

---

Prepared at LIMOS

Defended on December 11<sup>th</sup>, 2013

**Jury :**

*Reviewers :*

Pr. Mohammed Javeed ZAKI	Rensselaer Polytechnic Institute, USA
Pr. Abdoulaye BANIRÉ DIALLO	University of Quebec at Montreal, Canada
Pr. Jan RAMON	Katholieke Universiteit Leuven, Belgium

*Examiners :*

DR. David W. RITCHIE	INRIA, Nancy, France
DR. Jean SALLANTIN	LIRMM, Montpellier, France
DR. Jean-François GIBRAT	INRA, Jouy-en-Josas, France
Dr. Annegret WAGLER	University of Clermont-Ferrand II, France

*Advisor :*

Pr. Engelbert MEPHU NGUIFO	University of Clermont-Ferrand II, France
----------------------------	---



## Acknowledgment

First, I want to thank my advisor Prof. Engelbert Mephu Nguifo for his patience, support and efforts during the thesis that without them this thesis would not have been accomplished. Thanks for your precious advices on both professional and personal levels.

Special thanks to Prof. Mohammed J. ZAKI, Prof. Abdoulaye Baniré DIALLO and Prof. Jan RAMON for having accepted to review my thesis manuscript and for their kind efforts to comply with all the administrative constraints. My thanks go as well to DR. David W. RITCHIE, DR. Jean SAL-LANTIN, DR. Jean-François GIBRAT and Dr. Annegret WAGLER for having accepted to act as examiners of my thesis.

Thanks to all members of LIMOS Lab for having hosted me and allowed me working in comfortable conditions, with financial, academic and technical support. Special thanks to my office colleagues Jonathan PASSERAT-PALEMBACH, Faouzi JAZIRI, Sébastien CIPIÈRE and Pierre SCHWEITZER for the good years you were like brothers to me.

Most importantly, none of this would have been possible without the love and patience of my family; my father Mohammed DHIFLI, my mother Om Ezzine DHIFLI, my brothers and sisters, to whom this dissertation is dedicated to. They were always supporting me and encouraging me with their best wishes. Their support and care helped me overcome setbacks and stay focused on my graduate study. Their faith in me allowed me to be as ambitious as I wanted and helped me a lot through the past years.

On a more personal level, I want to thank my fiancée and future wife Kaouther OUERGHI, for her encouragement and for her faith in me that made me and still more and more ambitious. She was always there for me in the good moments as well as in the bad ones.

Special thank to Dr. Rabie SAIDI and Mohamed MOUSSAOUI for their valuable help in the accomplishment of the contributions.

At last but certainly not least, I would like to thank my friends, they were always there cheering me up and standing by me through both good and bad times.

Clermont-Ferrand, France, 2013  
Wajdi DHIFLI





# List of Figures

2.1	The different levels of the KDD process. . . . .	11
2.2	Using interestingness measures to mine interesting patterns. .	14
2.3	Pattern mining as pre-processing step in a pattern-based data mining framework. . . . .	15
2.4	The structure of DNA (right) and RNA (left). . . . .	20
2.5	A representation of the 3D-structure of the hemochromatosis protein. . . . .	20
2.6	Simplified process of transcription and translation. . . . .	21
2.7	The common structural scheme of amino acids. . . . .	21
2.8	The four levels of protein structure. . . . .	23
2.9	Exponential growth of the size of biological databases. . . . .	27
2.10	Triangulation example in a 2D-space. Left: Triangulation do meet the Delaunay condition. Right: Triangulation do not meet the Delaunay condition. . . . .	29
2.11	Example of protein 3D-structure (hemochromatosis) transformed into graph of amino acids (272 nodes and 1059 edges) using the main atom methods with $C_\alpha$ as the main atom and a distance thresholds $\delta \geq 7\text{\AA}$ . . . . .	31
3.1	Example of an unlabeled graph (a) and a labeled graph (b). .	36
3.2	Example of a subgraph (g) that is frequent in the graphs (a), (b) and (c) with a support of $\frac{3}{3} = 1$ . . . . .	37
3.3	Example of a candidates search tree in a BFS manner (a) and in a DFS manner (b). Each node in the search tree represents a subgraph candidate. . . . .	38
3.4	General schema of wrapper approaches for feature selection. .	43
3.5	General schema of embedded approaches for feature selection.	43
3.6	General schema of filter approaches for feature selection. . . .	44
4.1	An example of two representative unsubstituted patterns sampled from our experiments (from C-type lectin domains (DS3)). The figure also shows the corresponding position of each node of the subgraphs in the real protein 3D-structure. . . . .	65
4.2	Unsubstituted pattern selection framework. . . . .	66
4.3	An example of two substitutable subgraphs. . . . .	72
4.4	Rate of representative unsubstituted patterns $\Omega^*$ from all frequent subgraphs $\Omega$ depending on the substitution threshold ( $\tau$ ). .	79

4.5	Classification accuracy of the four protein datasets using naive bayes (NB) and all frequent subgraphs (gSpan+NB) then the representative unsubstituted ones with different substitution thresholds (UNSUBPATT+NB). . . . .	79
4.6	Classification accuracy by support vector machine (SVM). . .	80
4.7	Classification accuracy by decision trees (C4.5). . . . .	80
4.8	Distribution of patterns of DS1 for all the frequent subgraphs and for the representative unsubstituted ones with different substitution thresholds (left: standard arithmetic scale. Right: logarithmic scale.). . . . .	81
4.9	Distribution of patterns of DS2 for all the frequent subgraphs and for the representative unsubstituted ones with different substitution thresholds (left: standard arithmetic scale. Right: logarithmic scale.). . . . .	82
4.10	Distribution of patterns of DS3 for all the frequent subgraphs and for the representative unsubstituted ones with different substitution thresholds (left: standard arithmetic scale. Right: logarithmic scale.). . . . .	82
4.11	Distribution of patterns of DS4 for all the frequent subgraphs and for the representative unsubstituted ones with different substitution thresholds (left: standard arithmetic scale. Right: logarithmic scale.). . . . .	83
4.12	Classification accuracy comparison with Blast, Dali and pattern selection approaches. . . . .	84
5.1	An example of a graph of a chemical compound. . . . .	93
5.2	Distribution of subgraphs by size for the Enzymes dataset. . .	101
5.3	Distribution of subgraphs by size for the AIDS antiviral screen dataset. . . . .	102
5.4	Distribution of subgraphs by size for the Enzymes dataset using different values of $k$ (number of representatives). . . . .	102
5.5	Distribution of subgraphs by size for the AIDS antiviral screen dataset using different values of $k$ (number of representatives). . .	103
5.6	Runtime of clustering for TRS and the naïve approach with different number of clusters ( $k$ ). . . . .	104
5.7	Runtime of clustering for TRS and the naïve approach to select 100 representatives among different number of subgraphs. . . .	105
5.8	Runtime of clustering for TRS and the naïve approach to select 500 representatives among different number of subgraphs. . . .	105

---

5.9	Runtime of clustering for TRS and the naïve approach to select 100 representatives among 10000 subgraphs with variation of the number of graphs. . . . .	106
5.10	Runtime of clustering for TRS and the naïve approach to select 500 representatives among 10000 subgraphs with variation of the number of graphs. . . . .	107
A.1	PDB format. . . . .	115
A.2	FASTA format. . . . .	116
B.1	PGR general schema. . . . .	117
B.2	Parser. . . . .	118
B.3	PGR file. . . . .	119
B.4	Data . . . . .	119





# List of Tables

2.1	Alphabets of biological data. . . . .	21
2.2	The amino acids substitution matrix BLOSUM62. . . . .	25
3.1	Characteristics of Subgraph selection approaches according to different discription criterions. . . . .	56
4.1	Characteristics of the experimental datasets . . . . .	74
4.2	Number of frequent subgraphs ( $\Omega$ ), representative unsubstituted subgraphs ( $\Omega^*$ ) and the selection rate . . . . .	76
4.3	Accuracy, precision, recall (sensitivity), F-score and AUC of the classification of each dataset using NB coupled with frequent subgraphs (FSg) then representative unsubstituted subgraphs (USP) . . . . .	77
4.4	Number of subgraphs ( $\#SG$ ) and accuracy (Acc) of the classification of each dataset using NB coupled with frequent subgraphs (FSg) then representative unsubstituted subgraphs using Blosum62 ( $USP_{62}$ ), Blosum80 ( $USP_{80}$ ) and Pam250 ( $USP_{250}$ ) . . . . .	78
4.5	Runtime analysis of UNSUBPATT with different substitution thresholds . . . . .	85
5.1	Benchmark datasets . . . . .	98
5.2	Number of frequent subgraphs ( $\Omega$ ) extracted from each dataset . . . . .	99
5.3	Comparison of average information gain of the topological representative subgraphs (TRS) with those selected by the naïve approach (NA) and the initial set of all frequent subgraphs (FSG). . . . .	100
6.1	Characteristics of Subgraph selection approaches according to different discription criterions. . . . .	112



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Aims . . . . .	1
1.2	Context and motivations . . . . .	2
1.3	Contributions . . . . .	4
1.3.1	UNSUBPATT . . . . .	4
1.3.2	TRS . . . . .	4
1.4	Outline of the manuscript . . . . .	5
<b>I</b>	<b>Background and related works</b>	<b>7</b>
<b>2</b>	<b>Data mining and biological background</b>	<b>9</b>
2.1	Aims . . . . .	9
2.2	Data mining . . . . .	10
2.2.1	Knowledge discovery in databases . . . . .	10
2.2.2	Pattern mining in knowledge discovery . . . . .	11
2.2.3	Curse of dimensionality in pattern mining . . . . .	12
2.2.4	Measuring the quality of patterns . . . . .	13
2.2.5	Applications of patterns in data mining . . . . .	15
2.2.6	Applications of data mining . . . . .	18
2.3	Biological background : bioinformatics and biological data . .	18
2.3.1	Bioinformatics . . . . .	18
2.3.2	Biological data . . . . .	18
2.3.3	Protein structure . . . . .	21
2.3.4	From protein 3D-structures to protein graphs . . . . .	26
2.4	Conclusion . . . . .	31
<b>3</b>	<b>Related works</b>	<b>33</b>
3.1	Aims . . . . .	33
3.2	Graph mining . . . . .	34
3.3	Subgraph mining . . . . .	34
3.4	Frequent subgraph discovery . . . . .	35
3.4.1	Problem definition . . . . .	35
3.4.2	Candidates generation . . . . .	38
3.4.3	Frequent subgraph discovery approaches . . . . .	39
3.4.4	Variants of frequent subgraph mining: closed and max- imal subgraphs . . . . .	41

3.5	Feature selection . . . . .	41
3.5.1	Feature selection techniques . . . . .	42
3.5.2	Feature selection search strategy . . . . .	44
3.6	Feature selection for subgraphs . . . . .	44
3.6.1	Mining top-k subgraphs . . . . .	45
3.6.2	Clustering-based subgraph selection . . . . .	46
3.6.3	Sampling-based approaches . . . . .	48
3.6.4	Approximate subgraph mining . . . . .	49
3.6.5	Discriminative subgraph selection . . . . .	50
3.6.6	Other significant subgraph selection approaches . . . . .	53
3.7	Discussion . . . . .	55
3.8	Conclusion . . . . .	57

## II Contributions 59

<b>4</b>	<b>UnSubPatt: Mining representative unsubstituted graph patterns by means of substitution matrices</b>	<b>61</b>
4.1	Aims . . . . .	62
4.2	Introduction . . . . .	62
4.3	Mining representative unsubstituted patterns . . . . .	65
4.3.1	Background . . . . .	65
4.3.2	Preliminaries . . . . .	67
4.3.3	Algorithm . . . . .	70
4.3.4	Illustrative example . . . . .	72
4.4	Experiments . . . . .	73
4.4.1	Datasets . . . . .	73
4.4.2	Protocol and settings . . . . .	75
4.5	Results and discussion . . . . .	75
4.5.1	Empirical results . . . . .	76
4.5.2	Results using other substitution matrices . . . . .	77
4.5.3	Impact of varying the substitution threshold . . . . .	77
4.5.4	Smoothing the distribution of patterns . . . . .	81
4.5.5	Comparison with other approaches . . . . .	81
4.5.6	Runtime analysis . . . . .	84
4.6	Conclusion . . . . .	85
<b>5</b>	<b>TRS : Towards an efficient discovery of topological representative subgraphs</b>	<b>87</b>
5.1	Aims . . . . .	87
5.2	Introduction . . . . .	88

<b>Contents</b>	<b>xi</b>
5.3 Top-k topological representative subgraph selection . . . . .	90
5.3.1 Problem Statement . . . . .	90
5.3.2 Naïve approach . . . . .	90
5.3.3 Topological representative subgraph selection . . . . .	91
5.4 Experimental analysis . . . . .	97
5.4.1 Datasets . . . . .	97
5.4.2 Protocol and settings . . . . .	98
5.5 Results and discussion . . . . .	99
5.5.1 Empirical results . . . . .	99
5.5.2 Size-based distribution of patterns . . . . .	101
5.5.3 Runtime analysis . . . . .	103
5.6 Conclusion . . . . .	107
<b>6 Conclusion and future works</b>	<b>109</b>
6.1 Aims . . . . .	109
6.2 Summary of contributions . . . . .	110
6.2.1 UNSUBPATT . . . . .	110
6.2.2 TRS . . . . .	110
6.3 Discussion . . . . .	111
6.4 Ongoing works and prospects . . . . .	113
6.4.1 UNSUBPATT extensions . . . . .	113
6.4.2 TRS extensions . . . . .	114
<b>A Bioinformatics data formats</b>	<b>115</b>
A.1 PDB format . . . . .	115
A.2 FASTA format . . . . .	116
<b>B Protein Graph Repository</b>	<b>117</b>
B.1 Description . . . . .	117
B.2 How to use PGR ? . . . . .	117
B.2.1 Parser . . . . .	117
B.2.2 Repository . . . . .	118
<b>Bibliography</b>	<b>121</b>



# CHAPTER 1

## Introduction

---

### Contents

---

<b>1.1</b>	<b>Aims</b>	<b>1</b>
<b>1.2</b>	<b>Context and motivations</b>	<b>2</b>
<b>1.3</b>	<b>Contributions</b>	<b>4</b>
1.3.1	UNSUBPATT	4
1.3.2	TRS	4
<b>1.4</b>	<b>Outline of the manuscript</b>	<b>5</b>

---

## 1.1 Aims

In this chapter, we introduce the context and the main motivations of this thesis. We briefly present and discuss the proposed contributions. We also highlight the outline of each chapter in the manuscript as well as the appendices.



## 1.2 Context and motivations

This thesis is in the intersection of two of the most expanding research fields, namely data mining and bioinformatics. Data mining is one of the most active fields in computer science. It consists in analyzing complex data to extract useful information and transform them into understandable and more convenient format enabling and/or facilitating further use. The main goal of data mining is to provide useful tools and technical knowledge through algorithmic solutions for real world applications. Bioinformatics is an important application field for data mining. This is due to the complexity of biological processes and data that keep increasingly growing everyday. Manual work alone is unable to match the explosive growth of the amount of biological data. This rises an urgent need for automatic mining techniques to study these data.

Proteins are biological macromolecules that play crucial roles in almost every biological process. They are responsible in one form or another for a variety of physiological functions. Proteins are made of complex structure composed of a number of amino acids that are interconnected in space. The amino acids themselves are composed of a set of interconnected atoms. Thanks to both computational and biological advances we are witnessing these years, huge amounts of protein structures are currently available in online databases in computer analyzable formats. The biological importance of proteins, their complexity, and their availability in computer analyzable formats made us consider them as the main application data in this thesis.

Biologically speaking, the tertiary structure (shortly 3D-structure) of protein already contains its primary structure besides the connections between distant amino acids. It is the native form that controls the basic function of the protein. During the evolution some distantly related proteins may lose sequence homology while retaining some common folding. Hence, studying the tertiary structure of proteins is of great importance. A crucial step in the computational study of protein 3D-structures is to look for a convenient representation of their spatial conformation. Since a protein is composed of a set of connected amino acids, it can then be easily transformed into graphs where the amino acids are the graph nodes and their connections are the graph edges. Transforming protein 3D-structures into graphs enables using graph mining and more generally data mining techniques to study them.

Pattern mining is one of the most important tasks in data mining. The main purpose behind pattern mining is to find hidden relations and behaviors in data in order to better analyze them and to help in understanding the observed phenomena. Pattern mining has been extensively addressed during the last two decades for different types of patterns including association rules and itemsets. In the last few years, many efforts have been devoted to mine

patterns from graph data. This is not an easy task especially because of the combinatorial nature of graphs that makes the search space exponential.

Graph patterns can be in the form of properties (density, diameter, ...) or in the form of substructures. In this thesis, we are interested in patterns in the form of substructures and more specifically in the form of subgraphs. In this context, pioneer works were interested in mining subgraphs that are frequent in graph databases. This is mainly because of the benefit of antimonotonicity that offers the frequency measure. However, in the later studies, frequency taken by its own, is no longer enough to justify the importance of subgraphs. First, because many of the discovered frequent subgraphs are redundant or just useless for the user. Second, because of the high number of frequent subgraphs that hinder and even sometimes makes unfeasible further explorations. In the literature, this problem is sometimes referred to as *the curse of dimensionality* or *information overload*.

Several attempts have been made trying to resolve both mentioned issues by selecting only a small yet more interesting subset of subgraphs using interestingness measures that are defined according to the application needs. However, it is not always obvious to integrate the selection in the extraction process because most of the interestingness measures are neither monotonic nor antimonotonic.

Many approaches have been proposed for selecting interesting subgraphs, some of them are integrated in the extraction process, others perform the selection in post-processing. These approaches are investigated in Chapter 3. An interesting observation in existing subgraph selection approaches is that the *prior* information and knowledge about the application domain are often ignored. However, the latter provides valuable knowledge that may help building dedicated approaches that best fit the studied data. In this thesis, we propose two selection approaches for subgraphs. Both approaches aim to select representative subgraphs among the frequent ones in order to remove redundancy. Redundancy in frequent subgraphs is mainly caused by structural and/or semantic similarity, since most discovered subgraphs differ slightly in structure and may infer similar or even the same meaning. We attempt to overcome these shortcomings. Each of the proposed approaches addresses one type of redundancy, *i.e.*, the first approach focuses on semantic redundancy using the prior domain knowledge, while the second approach focuses on structural redundancy.

## 1.3 Contributions

### 1.3.1 UNSUBPATT

In existing subgraph selection approaches, the *prior* domain knowledge is often ignored. However, it can be exploited to build dedicated approaches that best fit the studied data. In our context, proteins evolve during the evolution where amino acids mutate from one type of amino acid into another through the action of DNA mutations. These mutations are quantified in the so-called *substitution matrices*. These matrices represent valuable domain knowledge that can be exploited. We propose UNSUBPATT (**Un**substituted **pat**terns), a subgraph selection approach that uses the substitution matrices to detect similarities between subgraphs. We show that this allows UNSUBPATT to detect similarities between subgraphs that current subgraph selection approaches ignore. This also enabled UNSUBPATT to select a small yet more representative and informative subset of subgraphs among frequent ones, enabling easier and more efficient further explorations. UNSUBPATT is unsupervised, thus, it can be used in any subgraph-based task. It is also worth noting that UNSUBPATT can be used for other sub-classes of patterns like trees and strings (represented as line graphs). Although UNSUBPATT is currently tested only on protein structures, this represents an immediate application example due to the availability of the substitution matrices. Indeed, UNSUBPATT can be used in any other application context whenever it is possible to define a matrix that quantifies similarities between the nodes labels.

### 1.3.2 TRS

The similarity between subgraphs in UNSUBPATT is purely semantic as it depends on the relations between nodes' labels, defined in the matrix. We introduce another subgraph selection approach, we term TRS (**T**opological **R**epresentative **S**ubgraphs), that focuses on the structural similarity rather than the semantic similarity. Existing works for structural subgraph selection are based on exact or approximate structural similarity. This similarity detection strategy is not efficient enough in many real-world applications. On one hand, the combinatorial nature of graphs makes looking for a possible matching between every pair of subgraphs computationally very costly. On the other hand, exact and even approximate structural similarity are not efficient enough to detect all similar subgraphs in real-world data. Indeed, exact structural similarity does not allow detecting similar yet slightly different subgraphs, and approximate structural similarity has the problem of threshold setting. A tight threshold prevent detecting similar subgraphs that slightly

differ in structure beyond the tolerance threshold. In contrast, a loose threshold will hinder the soundness of the selection because of false positives.

Unlike these approaches that look into every single detail, TRS follows a more meaningful selection by considering the overall structural similarity between subgraphs through a set of topological descriptors. This makes it easily extendable with any user-specified set of descriptors depending on the application and the sought information. TRS involves two steps. First, it encodes each subgraph into a topological description-vector containing the corresponding values for each one of the topological descriptors. Secondly, subgraphs with similar topological descriptions are clustered together and the central subgraph in each cluster is considered as a representative delegate. We show that TRS is able to select a set of topologically non-redundant and informative subgraph-delegates by considering hidden topological similarities between subgraphs that are ignored by current selection approaches. In addition, TRS is easily extendable with other types of descriptors and is not limited to biological data or to protein 3D-structures but can be used with any graph data. Moreover, TRS is unsupervised and can be used in any subgraph-based tasks.

## 1.4 Outline of the manuscript

The rest of this thesis is organized as follows. Chapter 2 presents the research field of this thesis namely, data mining as well as the application domain which is bioinformatics. It defines the basic notions and the preliminary concepts needed for the understanding of the rest of the thesis. We also focus on defining bioinformatics data and more precisely protein structures. We show the complexity of the latter and we review methods from the literature that allow transforming protein 3D-structures into graphs. We implemented these methods and made them available for public in a website which is presented at the end of the chapter and in the appendices.

In Chapter 3, we make a survey on related works over three levels. Since subgraph selection is always coupled with the extraction, in the first part we detail and discuss frequent subgraph mining algorithms in graph databases as well as existing approaches that address this task. The second part of the chapter focuses on the problem of feature selection in general and the last part of it reviews the most interesting subgraph selection approaches.

In Chapter 4, we propose a novel feature selection approach, termed UNSUBPATT, for selecting a subset of *representative unsubstituted* subgraphs among frequent ones. UNSUBPATT detects similarity between subgraphs by incorporating a specific domain knowledge which, in our context, consists of

the protein substitution matrices. Experimental evaluation of UNSUBPATT and comparison with other subgraph selection approaches from the literature are presented in the end of the chapter.

In Chapter 5, we propose another approach for subgraph selection, termed TRS. It selects a subset of *topological representative* subgraphs among the frequent ones. TRS focuses on the structural similarity to detect redundancy between subgraphs. It uses a set of user-defined measures to characterize the subgraphs, then it groups similar subgraphs into clusters to detect the representative subgraphs. We define a set of topological attributes then we use them to perform experimental analysis of TRS on a set of real and synthetic graph datasets.

Chapter 6 concludes the thesis by summarizing the proposed contributions and revealing ongoing works.

Appendix A describes the data format that we used in the experiments.

Appendix B gives a brief survey about PROTEIN GRAPH REPOSITORY (PGR) which is an online website that contains a tool for transforming protein 3D-structures into graphs and a repository mainly dedicated to protein graphs.

# Part I

## Background and related works



# Data mining and biological background

---

## Contents

<b>2.1</b>	<b>Aims</b>	<b>9</b>
<b>2.2</b>	<b>Data mining</b>	<b>10</b>
2.2.1	Knowledge discovery in databases	10
2.2.2	Pattern mining in knowledge discovery	11
2.2.3	Curse of dimensionality in pattern mining	12
2.2.4	Measuring the quality of patterns	13
2.2.5	Applications of patterns in data mining	15
2.2.6	Applications of data mining	18
<b>2.3</b>	<b>Biological background : bioinformatics and biological data</b>	<b>18</b>
2.3.1	Bioinformatics	18
2.3.2	Biological data	18
2.3.3	Protein structure	21
2.3.4	From protein 3D-structures to protein graphs	26
<b>2.4</b>	<b>Conclusion</b>	<b>31</b>

---

## 2.1 Aims

In this chapter, we present the preliminary concepts and the basic notions of the two main research fields of this thesis, namely data mining and bioinformatics. Specifically, we investigate the task of pattern mining, its main problem and how to resolve it. We also focus on defining bioinformatics data, precisely protein structures. We show their complexity and we review methods from the literature that allow transforming protein 3D-structures into graphs. This enables further analysis of protein structures using graph mining and more generally data mining techniques.



## 2.2 Data mining

In recent years, data mining has become one of the most active fields in computer science. This can be explained by the availability of increasingly huge amounts of data with an urgent need to analyze them. Besides, the huge advances we have witnessed in recent years in computational and storage technologies allow running greedy algorithms and analyzing more and more amounts of data.

Data mining is an interdisciplinary field in computer science. Different definitions have been given to data mining. One of the pioneer definitions was given in [Fayyad 1996]: *"Data mining is the application of specific algorithms for extracting patterns from data"*. In Wikipedia,<sup>1</sup> it is defined as the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. Han and Kamber defined it in [Han 2006] as: *"data mining refers to extracting or mining knowledge from large amounts of data"*. Zaki and Meira Jr. gave a similar definition in [Zaki 2014]: *"Data mining comprises the core algorithms that enable one to gain fundamental insights and knowledge from massive data"*. A cross view over the existing definitions allows us to simply consider data mining as the process of analyzing data to extract useful information and transform them into understandable and more convenient format, enabling and/or facilitating further use. Data mining often uses algorithms and techniques from statistics, artificial intelligence and databases, but may sometimes also involve techniques inspired from other domains such as physics, biology, chemistry, and so on.

### 2.2.1 Knowledge discovery in databases

Data mining is sometimes referred to as **K**nowledge **D**iscovery in **D**atabases or simply **KDD** [Fayyad 1997]. Yet, it only represents a part of the KDD process. In fact, KDD involves two other parts besides data mining, namely data pre-processing and data post-processing. Nevertheless, data mining may sometimes cover these two parts which makes it equivalent to KDD. Figure 2.1 illustrates the different levels of the KDD process.

It is necessary to differentiate between data, information and knowledge. From a computer science perspective, numbers, text, signals or in general any raw facts that can be processed by a computer is considered as data. Patterns, associations, and relationships among data can provide information. Thus, information is simply any meaning that could be understood from data.

---

<sup>1</sup>[http://en.wikipedia.org/wiki/Data\\_mining](http://en.wikipedia.org/wiki/Data_mining) (October 2013)

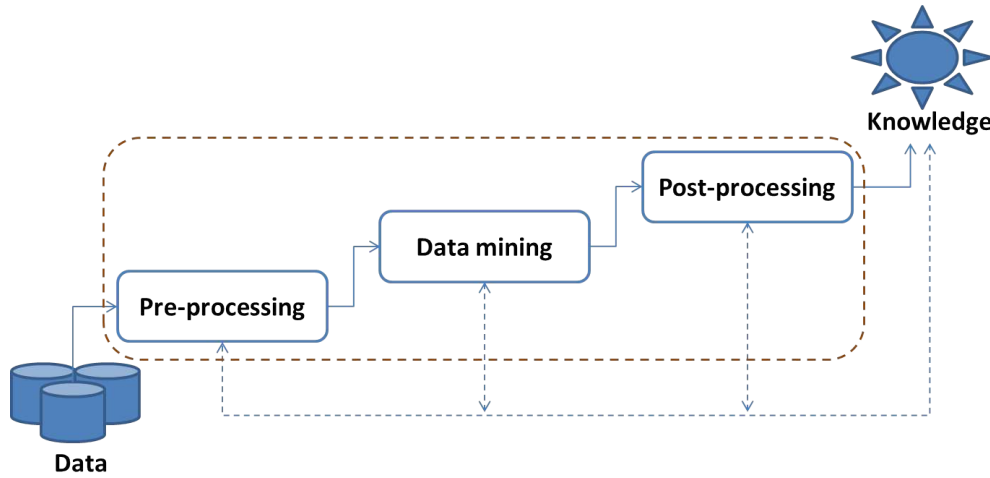


Figure 2.1: The different levels of the KDD process.

Based on the importance of information and its relevance to a given problem it can be considered as knowledge.

According to pioneer studies [Brachman 1996, Simoudis 1996, Mannila 1997, Fayyad 1997, Han 2006], the different steps of the KDD process can be defined as follows:

**Definition 1** (*Pre-processing*) It comprises all necessary procedures to prepare and parse data into adequate format for the data mining step. These procedures involves data cleaning to remove noise, data integration to combine multiple data source, data selection to extract subsets of data from the database that are concerned by the analysis, and data transformation to transform data into convenient formats that are required from data mining algorithms.

**Definition 2** (*Data mining*) It consists on applying mining techniques and computational methods in order to extract particular knowledge from data.

**Definition 3** (*Post-processing*) It consists on the evaluation, validation and interpretation of the knowledge discovered in the mining step. It can be performed based on algorithmic or visualization techniques.

### 2.2.2 Pattern mining in knowledge discovery

In statistics, the word population refers to the set or universe of the entities under study.

**Definition 4** (*Pattern*) A pattern (also referred as motif) consists generally on a feature which characterizes a given population.

Pattern mining is one of the core tasks and most important research fields in data mining. It consists on finding existing patterns in data. In fact, patterns can be in different forms ranging from simple patterns such as itemsets and association rules to more complex patterns such as sequences, trees and even to extremely complex patterns such as graphs and temporal or time evolving patterns. The main purpose behind pattern mining is to find hidden relations and behaviors in data in order to better analyze them and to help understanding the observed phenomena. Many of the pioneer works in pattern mining have been devoted to association rules. Indeed, the original motivation for finding association rules in data came from the desire to understand customers behavior in terms of the associations between purchased products in supermarket transactions. For example, how often do costumers buy *milk* and *sugar*, *i.e.*, find the support of the association rule "*milk*  $\rightarrow$  *sugar*". Such associations are very valuable to the supermarket owners as they may help, for instance, reorganizing the products positions according to the costumers preferences which may yield the increase of sales. Besides supermarket transactions, association rules and pattern mining in general are used in many other real application contexts such as identifying terrorists' activities and music information retrieval.

The identification of patterns in a given population is a hard task, since the miner has to answer to, at least, these questions in the first place:

- Which patterns are we seeking?
- How do they look like?
- How can we characterize them?
- How can we identify them?

Answering these questions before starting the pattern mining process is crucial, since each one of them highly affects the mining process.

### 2.2.3 Curse of dimensionality in pattern mining

A pattern can be identified in data based on one or several parameters such as its frequency in the population, its rarity or other user-defined criteria. The most common criterion used for pattern mining is frequency where the aim is to mine patterns that often occur in data. A pattern is considered as frequent if it occurs at least a minimum-number of times in the database. The minimum-number is user-defined and called *minimum support*. Mining frequent patterns stands under the assumption that patterns which frequently occur in data can be considered as features (events, relations, transactions, ...)

to characterize them. Indeed, it helps discovering interesting information and common behaviors in data. Yet, a serious problem that arises directly after mining frequent patterns consists in the huge number of discovered patterns that may reaches thousands and even millions. Such huge number of patterns may hinder or even makes unfeasible any further exploration. For example, it would make no sense to provide millions of patterns for visual inspection. In such case, instead of helping to resolve the problem, using frequent patterns will add a supplementary problem layer to resolve. This problem is referred to as the *information overload*. This problem has several side effects and consequences especially if the large set of patterns are used as attributes (dimensions) for further knowledge discovery tasks. This problem is referred to as the *curse of dimensionality*. It can be observed with simple patterns as well as with complex and sophisticated ones. Facing this problem, the main raised questions are:

- Are all the discovered patterns interesting?
- If not, how can we decrease the number of patterns without losing (at least approximately) any knowledge, such that only the significant patterns remain?

### 2.2.4 Measuring the quality of patterns

Resolving the problem of dimensionality is one of the biggest challenges in pattern mining, since a pattern mining algorithm will potentially generate a tremendous number of patterns especially in real world cases.

**Definition 5** (*Interesting pattern*) *A pattern is considered as interesting if it represents knowledge, i.e., it brings additional information regarding the mining problem or the information that is sought by the user.*

In order to overcome the dimensionality problem and to identify the truly *interesting* patterns that represent knowledge, some interestingness measures can be adopted to assist the pattern mining process. Interestingness measures are in the form of statistical functions. So far, there is no agreement on a formal definition of a universal measure that quantifies the importance of a pattern, or that allows distinguishing between the interesting patterns and the uninteresting ones. This is obvious, because in diverse applications and for different users the definition of the word *interesting* is relative and related to the goals. For this reason, a panoply of interestingness measures have been proposed in the literature.

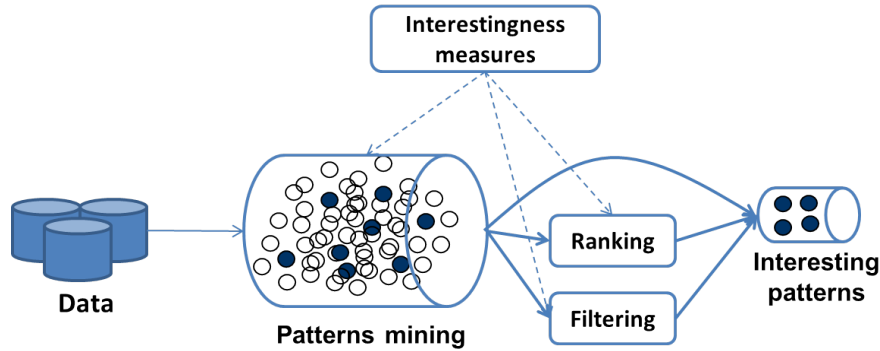


Figure 2.2: Using interestingness measures to mine interesting patterns.

Generally, the existing interestingness measures can be classified into two main groups, namely *objective measures* and *subjective measures* [McGarry 2005, Geng 2006]. Objective measures are usually based only on the raw data without requiring any knowledge about the user or the application. Most objective measures are based on statistical strength or characteristics of the patterns to assess their degree of interestingness. However, considering only objective interestingness measures may not allow highlighting the most interesting patterns. In contrast, subjective interestingness measures take into account both the domain knowledge and the user's beliefs. This is done by incorporating the background knowledge about the data and the user's knowledge and expectations during the mining process.

Both objective and subjective interestingness measures are used to assess the interestingness of patterns in two manners: ranking and filtering (see Figure 2.2). Ranking is performed by ordering the patterns according to their scores using the interestingness measures. Filtering is performed by eliminating all patterns that do not satisfy the criterion required by a measure. Since many measures quantify the interestingness of a pattern by a score, a user-defined threshold can be adopted to prune the irrelevant patterns such that only the ones satisfying the criteria remain. Filtering patterns can be performed during the mining process to avoid considering the irrelevant patterns by directly generating only the interesting ones. Or, it can be performed in post-processing where the interesting patterns are selected after generating the whole set. Computationally speaking, an efficient interestingness measure is the one that can be incorporated in the mining process and allows early termination to avoid walking over the whole pattern search space.

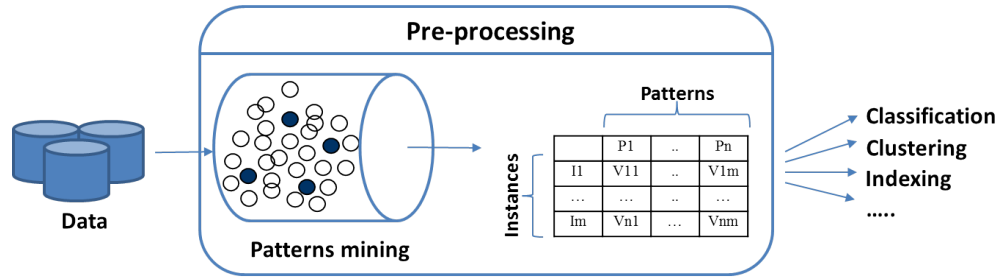


Figure 2.3: Pattern mining as pre-processing step in a pattern-based data mining framework.

### 2.2.5 Applications of patterns in data mining

Patterns are widely used in many data mining frameworks. In some applications, patterns can directly be considered as the sought information or even the sought knowledge, for instance for visual inspection. In other applications such as indexing and recommendation systems, patterns are used as a way to characterize the data under consideration and thus pattern mining will present only a pre-processing step for the pattern-based data mining framework. Classification and clustering are two of the major data mining applications where pattern mining is extensively used to transform raw data into pattern-based description that is accepted and processed by classification and clustering algorithms. In this context, patterns which occur in data are simply considered as *features* that characterize them. Patterns describing the data are also called *explanatory variables*. In the following, we define both classification and clustering since they are used later in the rest of the thesis. patterns which occur in data can be considered as features to characterize them.

#### 2.2.5.1 Classification

Classification is a supervised learning task. It consists of predicting the class label of an unknown object based on the observed labels of an already known set of objects. The set of known objects is usually referred as *training set*. Generally, a model is created by the classification algorithm, also called classifier, over the training data allowing to distinguish between the data classes.

There exists a panoply of classifiers that have been defined based on different techniques. The model building differs from one classifier to another. These classifiers range from simple ones like "One Rule" classifier to complex and sophisticated ones like "neural networks" and "support vector machines". In the following, we present three of the most known classifiers.

**Naive bayes** Naive bayes is a probabilistic classifier based on Bayes theorem [Stigler 1986, Wasserman 2010]. The idea is to use probability conditions to compute probability of each class. The predicted class is the one that maximizes the posterior probability. Naive bayes is called naive or simple because it assumes the independence of variables. Naive bayes is powerful and works well in many cases, however, in many applications, variables are not independent. Naive bayes is not suitable in such cases. Some variants of naive bayes were proposed attempting to overcome this drawback by assuming that variables can be related. Although this may contribute enhancing the results, it highly increases the computational cost. Thus, according to the *no-free-lunch theorem* [Wolpert 1995], no method is better than the others, every method counts and is appropriate for specific cases.

**Decision trees** Decision trees are ones of the most popular classifiers [Li 2008]. The most popular algorithms are ID3 [Quinlan 1986] and C4.5 [Quinlan 1993]. The goal consists in finding with the best possible accuracy the values taken by the variables to predict from a set of descriptors, *i.e.*, to best predict the class affiliation using the descriptors as features. The main idea is to consider the features as classification rules' conditions, then, try to find the best combinations of rules that best optimize the classifier prediction. Using the features, it constructs a tree-like model where the nodes are the features, the branches are the features' values, and the leaves are the predicted classes. Each path from the root node to a leaf present a classification rule.

**Support vector machines** Support Vector Machines, or shortly SVM [Vapnik 1995, Bi 2003], is a powerful classifier. SVM attempts to separate between positive and negative examples in the training set. Each example is represented by a feature vector. SVM seeks the hyperplane that best separates positive from negative examples, by ensuring that the margin between the closest positive and negative is maximal. New examples are encoded using the same features and predicted to belong to a class based on which side of the hyperplane.

### 2.2.5.2 Clustering

Clustering, also known as unsupervised classification, is the task of creating groups of objects based on one or more similarity criteria. The created groups of objects are also called clusters. Unlike classification, in clustering the class labels are unknown. Clusters presents homogeneous groups of objects that are created based on objects similarity. Thus, a good clustering tends to maximize similarity between objects within the same cluster (intra-cluster

similarity) while minimizing the overall similarity between different clusters (inter-cluster similarity).

There exists different clustering techniques depending on how the clustering is performed. Clustering techniques involves partitioning methods, hierarchical methods, density-based methods and others. In the following, we present some of the most known clustering algorithm. The list of existing clustering algorithms is not limited to the ones detailed below, namely k-Means [MacQueen 1967] and k-Medoids [Kaufman 1987], but it also involves other well known algorithms like EM [Dempster 1977], DBSCAN [Ester 1996], BIRCH [Zhang 1996], OPTICS [Ankerst 1999] and so on. We do only explain k-Means and k-Medoids as examples since clustering is not the main subject of this thesis, besides, k-Medoids is used later in Chapter 5.

**K-Means** K-Means is the most known clustering algorithm [MacQueen 1967, Jain 2010]. It is considered as a partitioning method. It takes as input a set of objects to be partitioned and a user-defined parameter  $k$  which corresponds to the number of clusters. The main goal of k-Means is to partition the objects into  $k$  clusters such that the intra-cluster similarity is maximized and the inter-cluster similarity is minimized according to a similarity (or inversely to a dissimilarity) function that computes distance between pairs of objects. K-Means proceeds as follows. It starts by randomly generating  $k$  fictive points as the clusters means (centroids). Then iteratively, it assigns each data point to the same cluster of the closest centroid. After assigning all the data points, the new mean point is computed and the assignment is reinitialized. K-Means iterates the cluster assignment and mean update until no change or local minima of criterion function converges.

**K-Medoids** K-Medoids [Kaufman 1987] is another partitioning method. It is considered as a variant of k-Means. It also accepts as input a set of objects to be partitioned and a user-defined number of clusters  $k$ . Then, it tries to partition the objects into  $k$  clusters following almost the same clustering procedure as k-Means. The main difference between k-Medoids and k-Means is that the latter defines the cluster's centers as fictive points, whereas, k-Medoids requires that the cluster's centroids be real points. This makes k-Medoids less sensitive to noise and outliers in the data. In addition, this makes it suitable for applications that looks for representative objects among data such that each centroid can be considered as the representative for all objects within the same cluster.



### 2.2.6 Applications of data mining

The main goal of data mining is to provide useful tools and technical knowledge through algorithmic solutions for real-world applications. Data mining can be useful in a variety of domains of application ranging from market analysis to chemistry and physics. One of the currently most expanding domains of application of data mining is bioinformatics. The main goal is to discover meaningful information and useful knowledge from biological data in order to help understanding biological phenomena such as the study of viruses [Diallo 2009], metabolic pathways [Morgat 2012], protein docking [Ritchie 2010], etc. In the following section, we try to define and detail the biological background as well as the biological data used in this thesis.

## 2.3 Biological background : bioinformatics and biological data

### 2.3.1 Bioinformatics

Bioinformatics is an interdisciplinary field. It can be simply defined by the application of computer science concepts and techniques to deal with biological data. Bioinformatics involves not only the collection, storage, prediction and analysis of molecules (nucleic acids, proteins, etc.) but also the development of tools for modeling biological systems through mathematical, statistical and computer science methods. Due to technological advances, bioinformatics has exponentially evolved during the past few years becoming one of the most expanding research fields nowadays. The emergence of bioinformatics did not only create a new application field for computer science, but also brought to biology many valuable benefits [Viari 2003]. Indeed, some tasks that used to require tremendous efforts and weeks or even months of lab work, do only need minutes or even seconds with the help of bioinformatics tools to perform the same task with often nearly similar quality. This is thanks to the high computational ability of current computer processors and to the algorithmic advances in the analysis and modeling of biological systems.

### 2.3.2 Biological data

Mainly, bioinformatics data revolve around three biological macromolecules namely DNA, RNA and protein. These three macromolecules are the essential component for all known forms of life.

## 2.3. Biological background : bioinformatics and biological data 19

---

### 2.3.2.1 DNA, RNA and proteins

**Nucleic data** Both DNA and RNA are called nucleic data.

- **DNA:** Deoxyribonucleic acid (shortly DNA) has a double helical twisted structure (as in Figure 2.4). Each side of the spiral is called a polymer. It is made of four parts called nucleotides: A (adenine), T (thymine), C (cytosine), and G (guanine). Both sides of the DNA are complementary, *i.e.*, whenever there is an edge of T, there is A in the corresponding position on the other side, and the same thing for G and C. DNA can be represented by a sequence of four letters, or bases. DNA is known to be the molecule that encodes the genetic instructions of all known living organisms and many viruses.
- **RNA:** Ribonucleic acid (shortly RNA, see Figure 2.4), is a long molecule but usually simple, except when it folds on itself. RNA perform multiple vital roles in the coding, decoding, regulation, and expression of genes. It differs chemically from DNA by containing the sugar ribose instead of deoxyribose and containing the base uracil (U) instead of thymine. Thus, the four RNA bases are A (adenine), U (uracil), C (cytosine), and G (guanine).

**Protein** Proteins are biological macromolecules formed by concatenation of 20 distinct amino acids (defined and detailed in Section 2.3.3)(see Figure 2.7 for the common structural scheme of amino acids) into long chains. They play crucial roles in almost every biological process. They are responsible in one form or another for a variety of physiological functions including enzymatic catalysis, binding, transport and storage, immune protection, control of growth, etc. A real example of a protein (the hemochromatosis protein) is illustrated in Figure 2.5 and more details about proteins are given in section 2.3.3.

### 2.3.2.2 The central dogma of molecular biology

The central dogma of molecular biology, detailed in [Crick 1958, Crick 1970], describes the biological macromolecules and the flow of genetic information between them (Figure 2.6). DNA is transcribed into RNA and the RNA is translated into proteins. The circular arrow around DNA denotes its ability to replicate which is the process of producing two identical copies from one original DNA molecule. From a computational perspective, these data can

---

<sup>2</sup>[http://en.wikipedia.org/wiki/Nucleic\\_acid\\_analogue](http://en.wikipedia.org/wiki/Nucleic_acid_analogue) (October 2013)

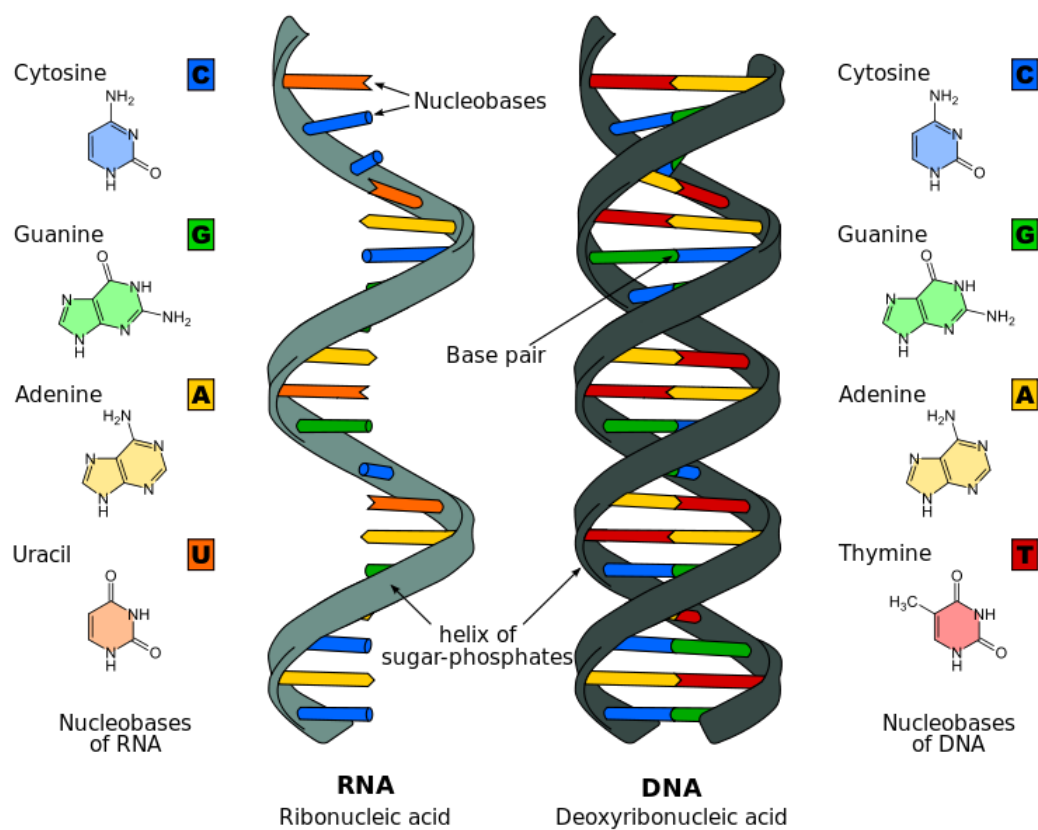


Figure 2.4: The structure of DNA (right) and RNA (left).<sup>2</sup>

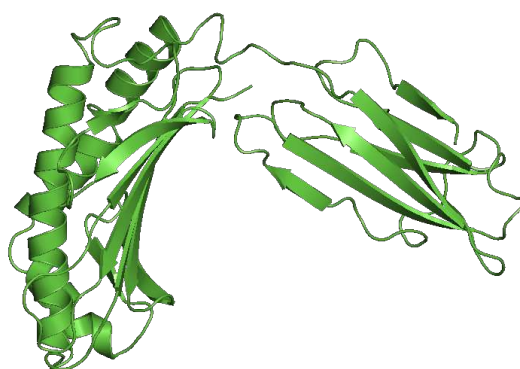


Figure 2.5: A representation of the 3D-structure of the hemochromatosis protein.

be seen as computer-readable structures defined within given alphabets as summarized in Table 2.1.

## 2.3. Biological background : bioinformatics and biological data 21

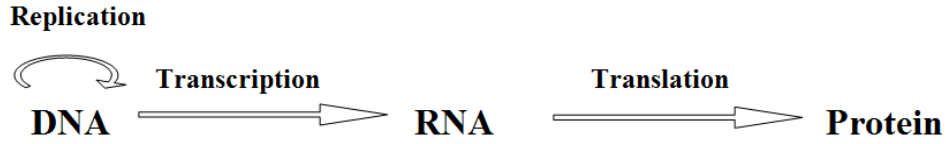


Figure 2.6: Simplified process of transcription and translation.

Table 2.1: Alphabets of biological data.

Type	Data	Alphabet
Nucleic	DNA	{A, T, C, G}
	RNA	{A, U, C, G}
Protein		{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y}

### 2.3.3 Protein structure

As in this thesis we are mainly interested in proteins as application data, here we further detail proteins, the different levels of protein structure as well as their chemical composition.

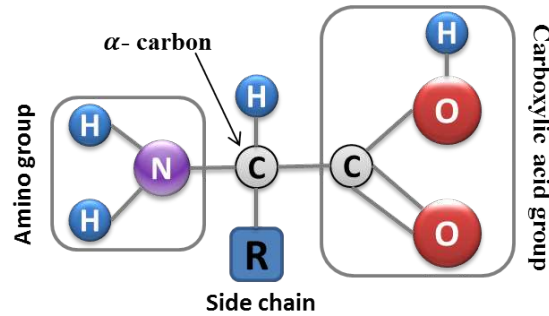


Figure 2.7: The common structural scheme of amino acids.

As previously mentioned, proteins are formed by the concatenation of a set of amino acids. These amino acids are within an alphabet of 20 distinct (see Table 2.1). Here is the list of amino acids where each one can be represented by a letter: alanine (A), cysteine (C), aspartic acid (D), glutamic acid (E), phenylalanine(F), glycine (G), histidine (H), isoleucine (I), lysine (K), leucine (L), methionine (M), asparagine (N), proline (P), glutamine (Q), arginine (R), serine (S), threonine (T), valine (V), tryptophan (W) and tyrosine (Y). All amino acids share a common structural scheme as illustrated in Figure 2.7. An amino acid is composed of a central carbon atom called  $C_{\alpha}$  and four chemical groups namely a hydrogen atom, an amino group (NH<sub>2</sub>), a carboxyl group (COOH) and a side chain or radical (R). The four chemical groups are

attached to the  $C_\alpha$  atom. It is the side chain that differentiates one amino acid to another and gives it its physico-chemical properties. The common parts between the amino acids compose the so called backbone [Branden 1991].

### 2.3.3.1 The four levels of protein structures

Amino acids constitute the building blocks of proteins. All amino acids of any protein are joined together by peptide bonds. Most proteins fold into unique three dimensional structures. However, it is possible to differentiate between four levels of protein structure as illustrated in Figure 2.8.

- **Primary structure:** The sequence of the amino acid residues in the chain is called the protein primary structure.
- **Secondary structure:** The chains of amino acids in the primary structure can fold to form complex three dimensional structures due to a combination of chemical interactions. These three dimensional fragments can take the form of one of three standard forms: a spiral conformation called  $\alpha$  – *helix*, a twisted pleated sheet called  $\beta$  – *sheet* and a *turn* where the polypeptide chain reverses its overall direction.
- **Tertiary structure:** The final folded state of a protein gives it its overall shape, *i.e.*, what is known as the protein tertiary structure (or simply protein 3D-structure). Precisely, it is formed by the spatial relations of the secondary structures such that even residues that are far away in the chain can be very close in the 3D-space.
- **Quaternary structure:** In reality, proteins are often composed of several sequences of amino acids. The quaternary structure of a protein consists on the combination of its sequences where each one has a primary, a secondary and a tertiary structure.

During the evolution, proteins go through changes. From one generation to another, the amino acids forming protein sequences are exposed to changes where they gradually mutate from one type of amino acid into another through the action of DNA mutations. Mutations of amino acids are quantified in the so-called *substitution matrix*.

### 2.3.3.2 Protein substitution matrices

A protein substitution matrix is a 20\*20 matrix where each value  $v$  between a pair of amino acids  $(x, y)$  presents the score of mutation of the amino acid  $x$  to the amino acid  $y$ , such that  $x, y \in [1..20]$ .

---

<sup>3</sup>[http://en.wikipedia.org/wiki/Protein\\_structure](http://en.wikipedia.org/wiki/Protein_structure) (October 2013)

### 2.3. Biological background : bioinformatics and biological data 23

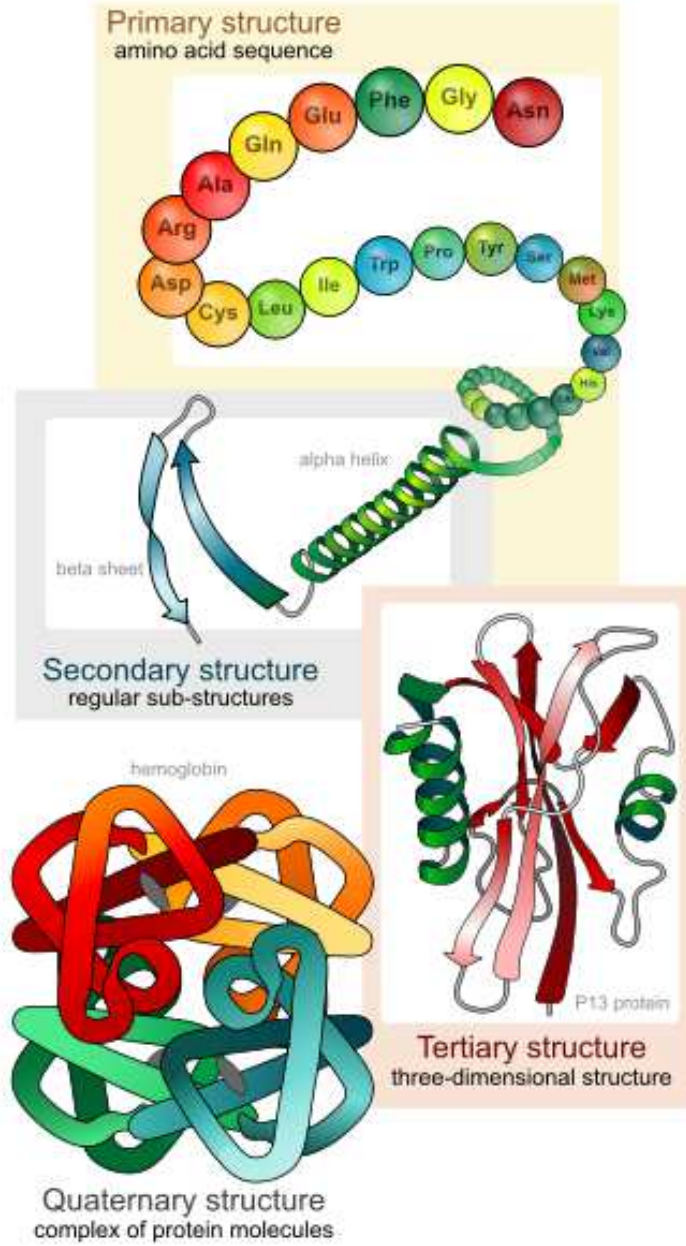


Figure 2.8: The four levels of protein structure.<sup>3</sup>

**Definition 6** (*Substitution matrix*) Given an alphabet  $\Sigma$  and a substitution matrix  $\mathcal{A}$  over  $\Sigma$ ,  $\forall l, l' \in \Sigma$ ,  $\mathcal{A}$  is defined as follows:

$$\mathcal{A}: \begin{cases} \Sigma^2 & \rightarrow [\perp, \top] \subset \mathbb{R} \\ (l, l') & \rightarrow s \end{cases}$$

The higher the value of  $s$  is, the more likely is the substitution of  $l'$  by  $l$ . Each

substitution score  $s$  is in  $[\perp, \top]$ . If  $s = \perp$  then the substitution is impossible, and if  $s = \top$  then it is certain. The values  $\perp$  and  $\top$  may appear or not in  $\mathcal{A}$ .

The most known protein substitution matrices are PAM [Dayhoff 1978] and BLOSUM [Henikoff 1992] :

**PAM matrices** PAM (Point Accepted Mutation) matrix was developed by Dayhoff [Dayhoff 1978]. This mutation matrix corresponds to a substitution accepted for 100 sites in a particular time of evolution, *i.e.*, a mutation that does not destroy the activity of the protein. This is known as a *one-percent-accepted-mutation matrix* (1PAM). If we multiply the matrix by itself a few times, we obtain a matrix XPAM that gives the probabilities of substitution for larger evolutionary distances under the assumption that repeated mutations would follow the same pattern as those in the 1PAM matrix. To be more easily used in sequence comparison programs, each XPAM matrix is transformed into a matrix of similarities PAMX called mutation matrix of Dayhoff [Dayhoff 1978]. This transformation is performed by considering the relative frequencies of mutation of amino acids and by taking the logarithm of each element of the matrix.

**BLOSUM matrices** A different approach was undertaken to highlight the substitution of amino acids. While PAM matrices derive from very similar proteins, here the degree of substitution of amino acids is measured by observing blocks of amino acids from more distant proteins. Each block present a short and highly conserved region. These blocks are used to group all segments of sequences having a minimum percentage of identity within their block. The frequency of substitution is deduced for each pair of amino acids then a logarithmic probability matrix called BLOSUM (BLOcks SUBstitution Matrix) is calculated over these frequencies. Every percentage of identity is a particular matrix. For instance, the BLOSUM62 matrix (see Table 2.2) is obtained using an identity threshold of 62%. Henikoff and Henikoff [Henikoff 1992] conducted such process from a database containing more than 2000 blocks.

Table 2.2: The amino acids substitution matrix BLOSUM62.

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	9	-1	-1	-3	0	-3	-3	-3	-4	-3	-3	-3	-3	-1	-1	-1	-1	-2	-2	-2
S	-1	4	1	-1	1	0	1	0	0	0	-1	-1	0	-1	-2	-2	-2	-2	-2	-3
T	-1	1	4	1	-1	1	0	1	0	0	0	-1	0	-1	-2	-2	-2	-2	-2	-3
P	-3	-1	1	7	-1	-2	-1	-1	-1	-1	-2	-2	-1	-2	-3	-3	-2	-4	-3	-4
A	0	1	-1	-1	4	0	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-2	-3
G	-3	0	1	-2	0	6	-2	-1	-2	-2	-2	-2	-2	-3	-4	-4	0	-3	-3	-2
N	-3	1	0	-2	-2	0	6	1	0	0	-1	0	0	-2	-3	-3	-3	-3	-2	-4
D	-3	0	1	-1	-2	-1	1	6	2	0	-1	-2	-1	-3	-3	-4	-3	-3	-3	-4
E	-4	0	0	-1	-1	-2	0	2	5	2	0	0	1	-2	-3	-3	-3	-3	-2	-3
Q	-3	0	0	-1	-1	-2	0	0	2	5	0	1	1	0	-3	-2	-2	-3	-1	-2
H	-3	-1	0	-2	-2	-2	1	1	0	0	8	0	-1	-2	-3	-3	-2	-1	2	-2
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5	2	-1	-3	-2	-3	-3	-2	-3
K	-3	0	0	-1	-1	-2	0	-1	1	1	-1	2	5	-1	-3	-2	-3	-3	-2	-3
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5	1	2	-2	0	-1	-1
I	-1	-2	-2	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4	2	1	0	-1	-3
L	-1	-2	-2	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4	3	0	-1	-2
V	-1	-2	-2	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4	-1	-1	-3
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6	3	1
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7	2
W	-2	-3	-3	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11



**PAMX, BLOSUMX: Which one is the best substitution matrix?**

The choice of protein substitution matrices depends on the type of experiments, the desired results, and the nature of data. Although many comparative studies have been conducted in this context [Yu 2005, Mount 2008, Brick 2008, Zimmermann 2010], no matrix is considered as the ideal one yet. However, it is clear from these studies that the matrices rather based on comparisons of sequences or 3D-structures usually give better results than those based primarily on the model of Dayhoff. For the most known substitution matrices,  $X$  is among {45, 52, 60, 80, 90} for BLOSUMX, and among {100, 120, 160, 200, 250} for PAMX. Higher BLOSUM matrices and lower PAM matrices are used to compare sequences that are relatively close and short while to compare more divergent and longer sequences, it is better to use lower BLOSUM or higher PAM. Most of current bioinformatics tools use BLOSUM62 (Table 2.2) as the substitution matrix selected by default.

**2.3.3.3 Availability of biological data**

The fast development we have witnessed in the past few years and still nowadays in both computer science and biological technologies, has highly facilitated the acquisition of biological data, transforming them into a computer readable format, and storing them into big databases. Indeed, this engendered the emergence of many online databases containing data that concerns different research areas from biology including genomics, proteomics, phylogenetics, metabolomics, and others. According to the Nucleic Acids Research (NAR) online molecular Biology Database Collection,<sup>4</sup> the number of databases registered in NAR has grown from 202 databases in the year 1999 to 1512 in 2013. This increase has not only been noticed on the number of databases but also on their sizes. This fact is illustrated in Figure 2.9 which shows the exponential increase in size of some of the most known biological databases.

Manual work alone is unable to match the explosive growth of the amount of biological data. This arises an urgent need for new bioinformatics tools and new data mining algorithms for automatic analysis and knowledge retrieval.

**2.3.4 From protein 3D-structures to protein graphs**

For many years, proteins have been mainly studied based on their primary structure. This is because the primary structure is more simple to represent than all the other structures since it can be seen as string of characters where each character represents one corresponding amino acid from the chain. In addition, there has been a huge gap between the number of unique protein

<sup>4</sup><http://www.oxfordjournals.org/nar/database/a/>

### 2.3. Biological background : bioinformatics and biological data 27

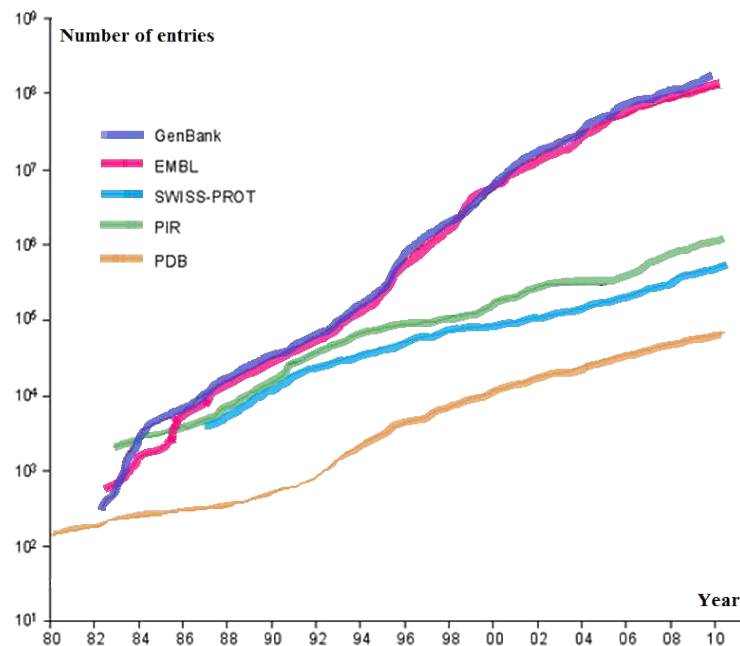


Figure 2.9: Exponential growth of the size of biological databases.

primary structures in databases and that of protein 3D-structures. Moreover, in that stage, the algorithmic advances allow to efficiently handle and deal with strings of characters.

In recent years, increasing efforts have been devoted to deal with protein 3D-structures. This is due to the huge number of currently available protein 3D-structures in online databases. Besides, the algorithmic and technological advances have allowed to run greedy algorithms and to better handle more complex data. Biologically speaking, the tertiary structure of protein is more important than its primary structure. First, because the tertiary structure already contains the primary structure besides the connections between distant amino acids. In addition, the tertiary structure is the native form that controls the basic function of the protein. Moreover, during the evolution some distantly related proteins may lose sequence homology while retaining some common folding.

There exists online databases allowing the acquisition of protein structures in computer readable formats. Protein Data Bank [Berman 2000] is the most known one.

### 2.3.4.1 PDB: Protein Data Bank

One of the most known databases is the [Protein Data Bank](#)<sup>5</sup> (shortly PDB) [[Berman 2000](#)] which is a free online repository of information about the 3D-structures of large biological molecules, including proteins and nucleic acids. The PDB was created in 1971 at Brookhaven National Laboratory, and is continuously expanding. By the end of September 2013, it contains already 94540 structures, and the repository gets updated every week. In addition, the PDB website allows users to perform simple and complex queries on the data, analyze them, and visualize the results.

Protein structures are available in the PDB website in a special data format also called *PDB*. In fact, the PDB file is simply a textual format describing the coordinates of atoms of a molecule in the 3D-space (see [Section A.1](#) for more details).

### 2.3.4.2 Parsing protein 3D-structures into graphs

A crucial step in the computational study of protein 3D-structures is to look for a convenient representation of their spatial conformation. Since a protein can be seen as a set of connected elements (amino acids and atoms), it can then be easily transformed into a graph where the elements are the graph nodes and the connections are the graph edges. In most existing works, proteins are transformed into graphs of amino acids where each one of the latter is represented by a node in the graph and the graph edges represent the connections between the amino acids.

**Transformation techniques:** Some approaches have been proposed in the literature for transforming protein 3D-structures into graphs of amino acids [[Saidi 2009](#)]. These approaches use different techniques. In the following, we present the most known approaches. In all these approaches, nodes of the graphs represent the amino acids. However, they differ in the way of considering the edges in attempt to reflect the truly existing interactions. Some of them express the edges by the strength of interaction between amino acids' side chains, while, others express the edges based on the distance between pairs of amino acids.

- **Triangulation** Triangulation is used to transform an object, represented by a set of points in a plane or in a 3D-space, into a set of triangles. It is possible to have multiple triangulation for the same object. The Delaunay triangulation [[Delaunay 1934](#)] is a special way

---

<sup>5</sup><http://www.rcsb.org/pdb/> (August 2013)

## 2.3. Biological background : bioinformatics and biological data 29

of triangulation. It was used to build protein graphs in several works [Bostick 2004, Huan 2005, Stout 2008]. The main idea is to consider the amino acids as a set of points in the space, then to iteratively try to create tetrahedrons such that no point is inside the circum-sphere<sup>6</sup> of any tetrahedron, *i.e.*, empty spheres (see the example in Figure 2.10).

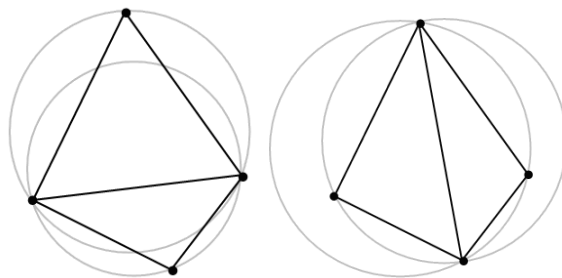


Figure 2.10: Triangulation example in a 2D-space. Left: Triangulation do meet the Delaunay condition. Right: Triangulation do not meet the Delaunay condition.

- **Main Atom** This is the main approach used in the literature. The main idea is to abstract each amino acid only to a main atom  $M_A$  of it. This main atom can be real, like the  $C_\alpha$  or the  $C_\beta$  atoms, or fictive, like the amino acid centroid or the side chain centroid [Lovell 2003, Huan 2005]. Two nodes representing the amino acids  $u$  and  $v$  are linked by an edge  $e(u, v) = 1$ , if the euclidean distance between their two main atoms  $\Delta(M_A(u), M_A(v))$  is below a threshold distance  $\delta$ . Formally:

$$e(u, v) = \begin{cases} 1, & \text{if } \Delta(M_A(u), M_A(v)) \leq \delta \\ 0, & \text{otherwise} \end{cases} \quad (2.1)$$

In the literature, many works used this method basically with  $C_\alpha$  atom and with usually  $\delta \geq 7\text{\AA}$  on the argument that  $C_\alpha$  atoms define the overall shape of the protein conformation [Huan 2005].

- **All Atoms** Some extensions have been made to the main atom method. For instance in [Saidi 2009], authors proposed *all atoms* where instead of considering the distances only between the main atoms of amino acid,

---

<sup>6</sup>A tetrahedron is a polyhedron composed of four triangular faces that meet at each corner. A circum-sphere of a polyhedron is a sphere that contains the polyhedron and touches each of its vertices.

they consider it between all the atoms of the amino acids ( $A_A$ ). Formally:

$$e(u, v) = \begin{cases} 1, & \text{if } \Delta(A_A(u), A_A(v)) \leq \delta \\ 0, & \text{otherwise} \end{cases} \quad (2.2)$$

**Discussion** Here, we discuss the above mentioned transformation techniques of protein 3D-structures into graphs. We assume that the correctness of each technique is measured by its ability to reflect in the edges of the graph it generates, the really existing links in the protein. The Delaunay triangulation method suffers from two main drawbacks because of the empty circum-spheres condition. First, we can find many false links between very far nodes in the protein especially at the surface of the protein where the circum-spheres get easily out of the cloud of atoms. Second, the empty sphere condition does not allow a node to make connection with any other one outside of its tetrahedron sphere. This makes it omit many edges even in the presence of real interactions.

The main atom method suffers a drawback. Since it abstracts the amino acids into one main atom, it may omit possible edges between other atoms in the amino acids that are more close than the main atoms. Moreover, in the case of considering the centroids of the amino acids as the main atoms, it may also suffer from two problems. In the case where the amino acids are big, if the centroids of the amino acids are farther than the given distance threshold then they will be considered with no links while a real connection could be established between other close atoms. In the case where the amino acids are small, if the distance between the centroids of the amino acids is smaller than the given distance threshold then they will be considered as connected while they can be disconnected in reality. To overcome main atom drawbacks, all atoms method considers theoretically the distance between all the atoms in the amino acids, this highly increases the complexity of the execution time. Besides, among the heuristics the authors proposed to alleviate the complexity of their approach, they do consider only the distance between the centroids of the side chains of amino acids to decide whether they are connected or not, without considering their chemical properties. This may engender many false edges.

As biologically speaking the  $C_\alpha$  atoms define the overall shape of the protein conformation [Huan 2005], we choose to use the main atom method in the experiments we conduct, using  $C_\alpha$  as the main atom. A real example of a protein 3D-structure (the hemochromatosis protein) transformed into a graph is illustrated in Figure 2.11. Although using such representation may omit some edges and contains some false ones, it opens new challenges and

perspectives by allowing the use of graph mining techniques to study proteins. Besides, it enables extending previous works from the literature on primary sequences like pattern discovery [Kleywegt 1999, Doppelt 2007].

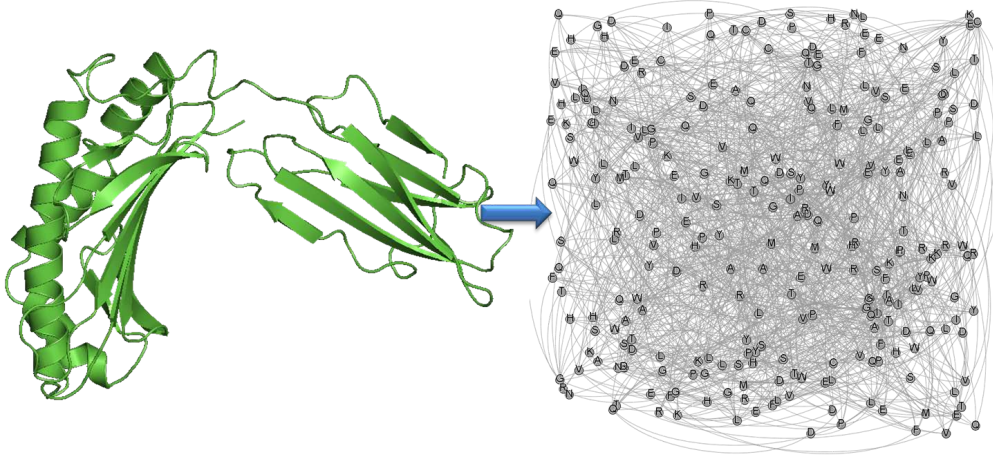


Figure 2.11: Example of protein 3D-structure (hemochromatosis) transformed into graph of amino acids (272 nodes and 1059 edges) using the main atom methods with  $C_\alpha$  as the main atom and a distance thresholds  $\delta \geq 7\text{\AA}$ .

**PGR: protein graph repository** We have implemented the mentioned transformation methods from protein 3D-structures to graphs and other ones in a jar file. This file is used as the core program of a web repository for proteins graphs termed Protein Graph Repository (shortly PGR) [Dhiifi 2010]. PGR accepts protein PDB files as input and outputs graphs of amino acids under several format (see Appendix B for more details).

## 2.4 Conclusion

In this chapter, we presented the main research area of this thesis, namely data mining. Particularly, we focused on one of the core tasks in data mining and more generally in knowledge discovery which is the task of pattern mining. We investigated the curse of dimensionality which is one of the main problems in pattern mining that consists on the exponential number of patterns and we showed how to resolve it using the interestingness measures. We also presented the main domain of application of this thesis namely bioinformatics and more precisely protein 3D-structures. We showed that proteins are complex data that need to be analyzed and mined using automatic methods because of the large amounts of data that keep increasing everyday. We reviewed the existing

methods to transform protein 3D-structures into graphs which enables using graph mining and more generally data mining techniques to study them. In the next chapter, we will review existing works in the area of pattern mining over graph data, and we will focus on the problem of feature selection for graph patterns as a way to tackle the curse of dimensionality.

## CHAPTER 3

# Related works

---

### Contents

---

<b>3.1</b>	<b>Aims</b>	<b>33</b>
<b>3.2</b>	<b>Graph mining</b>	<b>34</b>
<b>3.3</b>	<b>Subgraph mining</b>	<b>34</b>
<b>3.4</b>	<b>Frequent subgraph discovery</b>	<b>35</b>
3.4.1	Problem definition	35
3.4.2	Candidates generation	38
3.4.3	Frequent subgraph discovery approaches	39
3.4.4	Variants of frequent subgraph mining: closed and maximal subgraphs	41
<b>3.5</b>	<b>Feature selection</b>	<b>41</b>
3.5.1	Feature selection techniques	42
3.5.2	Feature selection search strategy	44
<b>3.6</b>	<b>Feature selection for subgraphs</b>	<b>44</b>
3.6.1	Mining top-k subgraphs	45
3.6.2	Clustering-based subgraph selection	46
3.6.3	Sampling-based approaches	48
3.6.4	Approximate subgraph mining	49
3.6.5	Discriminative subgraph selection	50
3.6.6	Other significant subgraph selection approaches	53
<b>3.7</b>	<b>Discussion</b>	<b>55</b>
<b>3.8</b>	<b>Conclusion</b>	<b>57</b>

---

## 3.1 Aims

In this chapter, we present works in the literature that are relevant to this thesis. These works are presented over three levels. As many of the subgraph



selection approaches as well as the ones proposed in this thesis perform the selection over the set of frequent subgraphs, the first part of the chapter details and explains the problem of frequent subgraph mining in graph databases and lists the main used approaches to address this task. In the second part of the chapter, we present in general the problem of feature selection and discuss the different possible techniques and search strategies. In the last part, we review a panoply of the most interesting subgraph selection approaches. We try to partition these approaches into different groups depending on the selection strategy and or the type of the selected subgraphs. A descriptive table presenting the characteristics of the mentioned subgraph selection approaches is presented as well.

## 3.2 Graph mining

Graphs are one of the most powerful structures to model complex data [Cook 2006]. In fact, any data composed of entities having relationships can be represented by a graph where the entities will be seen as the graph nodes and the relationships as the graph edges. Recently and thanks to the increasingly cheaper cost of storage devices and the availability of high processing power, graphs are becoming ubiquitous. They are increasingly used in the modeling and the analysis of many real world applications such as the world wide web, blogs, cell phone communications, XML documents, and even electronic circuits. In chemoinformatics and bioinformatics, graphs are used to model various types of molecules and biological data such as chemical compounds, gene and metabolic networks, protein structures and protein-protein interaction networks. Graphs are also used in the analysis of social networks such as Facebook and Google+, where graphs represents networks of connected users and are used to understand phenomena such as rumors propagation and criminal networks or to predict links or to detect central users.

One of the most powerful techniques to analyze and study graph data is to look for interesting subgraphs among them. Subgraphs are said to be interesting if they obey to one or different constraints. These constraints can be structural and topological, based on frequency, coverage, discrimination or even semantic if the graphs are labeled.

## 3.3 Subgraph mining

One of the main and most challenging tasks in graph mining is to look for recurrent substructures, *i.e.*, to extract frequent subgraphs [Cheng 2010]. In fact, there exists two types of frequent subgraph discovery namely

- From a single large graph: In this case, we wish to determine all subgraphs that occur at least a certain number of times in a one large graph (e.g., the World Wide Web graph).
- From a database of many graphs: In this case, we have a database of graphs (e.g., a family of protein 3D-structures represented by graphs) and we wish to determine all subgraphs that occur at least in a certain number of graphs of the database.

In different applications, we may be interested in different kinds of subgraphs, such as subtrees, cliques (or complete graphs), bipartite cliques, dense subgraphs, and so on. These subgraphs are used later as patterns to describe the data under consideration. Indeed, they may represent, for example, communities in social networks, hubs and authority pages on the WWW, clusters of proteins involved in similar biochemical functions in protein-protein interaction networks, and so on. But in the most common case, subgraphs are mined from data based on their frequency.

## 3.4 Frequent subgraph discovery

The problem of frequent pattern mining has been widely addressed in data mining. Yet, in the case of graph data, mining frequent patterns is more challenging mainly because of the combinatorial nature of graphs [Zaki 2014]. Indeed, in the case of graphs the process of determining support is different. As in this thesis we are more interested in the mining of frequent subgraphs from a graph database, this subsection defines and gives the formal statement of the problem of frequent subgraph discovery in graph databases.

### 3.4.1 Problem definition

Let  $\mathcal{G}$  be a graph database. Each graph  $G = (V, E)$  of  $\mathcal{G}$  is given as a collection of nodes  $V$  and edges  $E$ . We denote by  $|V|$  the number of nodes of  $G$  (also referred as the graph order) and by  $|E|$  the number of edges of  $G$  (also called the graph size). If two nodes  $u$  and  $v \in V$  and  $\{u, v\} \in E$  then  $u$  and  $v$  are said to be adjacent nodes. The nodes and edges of  $G$  can be labeled within an alphabet  $\Sigma$  such that  $G$  becomes  $G = (V, E, \Sigma, L)$  where  $\Sigma = \Sigma_V \cup \Sigma_E$  and  $L$  is the label function that maps a node or an edge to a label (Figure 3.1 shows an example of an unlabeled graph (a) and a labeled graph (b)).  $G$  is called a labeled graph and the labels of nodes and edges are denoted respectively by  $L(u)$  and  $L\{u, v\}$ .

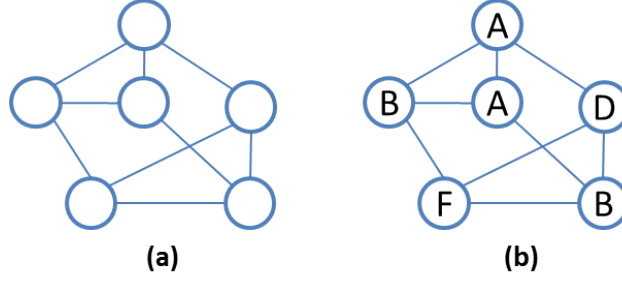


Figure 3.1: Example of an unlabeled graph (a) and a labeled graph (b).

**Definition 7** (*Graph isomorphism*) A graph isomorphism exists between two graphs  $G = (V, E, \Sigma, L)$  and  $G' = (V', E', \Sigma', L')$  if there is a bijective function  $f: V \rightarrow V'$  such that:

- $\forall u, v \in V : \forall \{u, v\} \in E \Leftrightarrow \{f(u), f(v)\} \in E'$
- $\forall v \in V : L(v) = L'(f(v))$
- $\forall \{u, v\} \in E : L\{u, v\} = L'\{f(u), f(v)\}$

where  $L$  and  $L'$  are respectively the label functions of  $G$  and  $G'$ .

**Definition 8** (*Subgraph isomorphism*) A labeled graph  $G$  is subgraph of another labeled graph  $G'$ , denoted by  $G \subseteq G'$ , if there exists an injective function  $f: V \rightarrow V'$ , such that:

- $\forall u, v \in V : \forall \{u, v\} \in E \rightarrow \{f(u), f(v)\} \in E'$
- $\forall v \in V, L(v) = L'(f(v))$
- $\forall \{u, v\} \in E : L\{u, v\} = L'\{f(u), f(v)\}$

Under these conditions, the function  $f$  is called an embedding of  $G$  in  $G'$ ,  $G$  is called a subgraph of  $G'$ , and  $G'$  is called a supergraph of  $G$ .

**Definition 9** (*Frequent subgraph*) Given a subgraph  $g$ , a graph database  $\mathcal{G}$ , and a minimum frequency threshold  $\tau$  (minimum support), let  $\mathcal{G}_g$  be the subset of  $\mathcal{G}$  where  $g$  appears (i.e.,  $g$  has a subgraph isomorphism in each graph in  $\mathcal{G}_g$ ). The number of graphs where  $g$  occurs is denoted by  $|\mathcal{G}_g|$ . The subgraph  $g$  is considered as frequent if:

$$\text{support}(g) = \frac{|\mathcal{G}_g|}{|\mathcal{G}|} \geq \tau \quad (3.1)$$

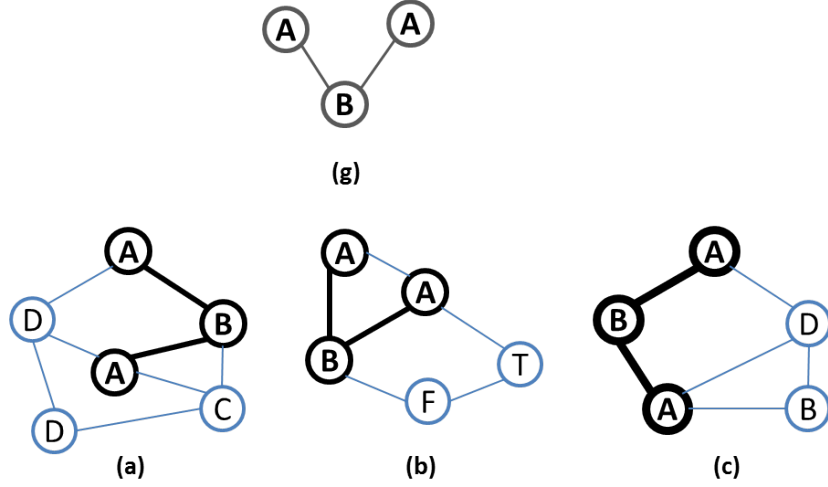


Figure 3.2: Example of a subgraph (g) that is frequent in the graphs (a), (b) and (c) with a support of  $\frac{3}{3} = 1$ .

Figure 3.2 shows an example of a subgraph (g) that is frequent in the graphs (a), (b) and (c) with a support of  $\frac{1+1+1}{3} = 1$ , i.e., (g) is frequent for all minimum frequency threshold  $\tau \in [0..1]$ .

Mainly, frequent subgraph mining is performed over two steps. The first step consists on generating the candidate subgraphs. In the second step, each one of the subgraph candidates is checked for frequency. Among the different kinds of frequent pattern mining, frequent subgraph mining is more challenging. Indeed, the challenge in frequent subgraph mining comes from the costly graph isomorphism and subgraph isomorphism which are known to be NP-complete in their generalization. In the first step, the graph isomorphism is used to avoid the generation of duplicate subgraph candidates. In the second step, subgraph isomorphism checks are needed in order to determine the support of each subgraph candidate in all graphs in the database.

Since graphs are combinatorial by nature, the number of possible extensions of subgraph candidates is exponential. Thus, it is of an urgent need to limit the search space.

**Property 1 (Anti-monotonicity)** *All subgraphs of a frequent subgraph are themselves frequent. Consequently, all supergraphs of an infrequent subgraph are infrequent.*

This property is very useful in pruning the candidates search space, since it says that there is no need to make further extensions if the subgraph candidate is no longer frequent.

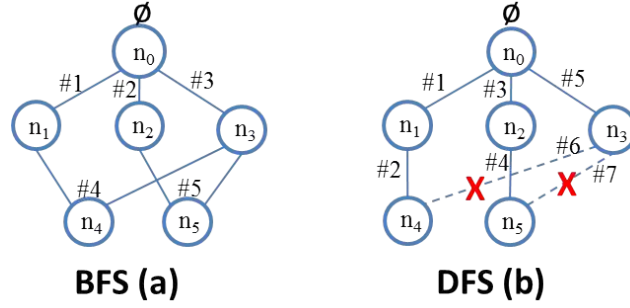


Figure 3.3: Example of a candidates search tree in a BFS manner (a) and in a DFS manner (b). Each node in the search tree represents a subgraph candidate.

### 3.4.2 Candidates generation

The generation of candidates is an essential step for the discovery of frequent subgraphs. Generally, a candidate search space is iteratively constructed, in a tree-like form, by extending the subgraph candidates where each node in the tree represents a subgraph candidate and each child node of the latter represents one of its possible extensions, *i.e.*, one of its supergraphs. There are two main techniques for generating candidates, namely Breadth First Search (BFS) and Depth First Search (DFS) (see the illustrative example below). The first approach (BFS) follows a level-wise strategy where candidates of size  $k+1$  are generated by merging and fusing candidates of size  $k$ . That is to say, that no candidate of size  $k+1$  is generated before generating and exploring all candidates of size  $k$ . In a DFS strategy, a subgraph candidate is iteratively checked for frequency then arbitrarily extended until it is no longer frequent or no further extension is possible. Most recent algorithms prefer DFS over BFS because it uses less memory and is faster since it avoids the costly merge and fusion of subgraphs used in BFS.

**Illustrative example** Figure 3.3 shows an illustrative example of a candidate search tree in a BFS manner (a) and in a DFS manner (b). Each node in the search tree represents a subgraph candidate. Since BFS (Figure 3.3 (a)) follows a level-wise strategy, the tree is constructed using the following order:

1. Level 1:  $n_0$
2. Level 2:  $n_1, n_2, n_3$
3. Level 3:  $n_4$  (constructed from  $n_1$  and  $n_3$ ),  $n_5$  (constructed from  $n_2$  and  $n_3$ ).

The DFS (Figure 3.3 (b)) follows the same branch until the bottom of the tree before visiting the other branches. Supposing that nodes on the right are visited before those on the left, candidates in the tree are constructed using the following order:

- $n_0, n_1, n_4, n_2, n_5, n_3$ , ( $n_4$ : but pruned for duplication since already generated from  $n_1$ ), ( $n_5$ : but pruned for duplication since already generated from  $n_2$ ).

### 3.4.3 Frequent subgraph discovery approaches

Many approaches for frequent subgraph mining have been proposed [Jiang 2013, Lakshmi 2012, Krishna 2011]. A pioneer work is [Cook 1994], where authors proposed an approximate and greedy search algorithm named *SUBDUE* for discovering frequent graph patterns based on a minimum description length and background knowledge. Other works have been proposed based on the principles of artificial intelligence, like *WARMR* [King 2001] and *FARMER* [Nijssen 2001]. They successfully mined frequent subgraphs from chemical compounds data. Although these approaches allow to completely discover all frequent subgraphs, they suffer from the high consumption in terms of time and computational resources. In addition, the discovered subgraphs are semantically very complex since the graphs were initially transformed into datalog facts.

Besides these studies, there exists two main categories of the approaches of frequent subgraph discovery namely: the apriori-based approaches and the pattern-growth approaches.

#### 3.4.3.1 Apriori-based approaches

Generally, apriori-based approaches start from subgraphs of small sizes. Then, in a bottom-up manner, they generate subgraph candidates by adding a node or an edge to an existing frequent subgraph. The main idea behind apriori-based approaches is that subgraph candidates of size  $k + 1$  are generated by means of a join operation on two frequent subgraphs of size  $k$  having a common subgraph of size  $k - 1$ . Thus, in order to be able to generate subgraph candidates of level  $k + 1$ , all subgraphs of size  $k$  have to be already generated. Hence, the name apriori-based approaches. Consequently, all apriori-based approaches have to use the BFS strategy since they follow a level-wise candidate generation.

The main algorithms that have been proposed in this category are AGM [Inokuchi 2000], FSG [Kuramochi 2001] and DPMine [Vanetik 2002]. AGM and FSG are very similar, but the main difference between them is that AGM

generates a candidate by extending a frequent subgraph with a node. While, FSG generates a candidate by extending a frequent subgraph with an edge. DPMine uses edge-disjoint paths as the expansion units for candidate generation. It starts by identifying all frequent paths, then, all subgraphs with two paths. After that, it starts generating subgraphs with  $k$  paths by merging pairs of frequent subgraphs of  $k-1$  paths having  $k-2$  paths in common.

### 3.4.3.2 Pattern-growth approaches

Pattern-growth approaches extend an already discovered frequent subgraph by adding an edge in every possible position. Adding an edge may result adding a new node. Extensions are recursively performed until no more frequent subgraph is generated. In contrast to apriori-based approaches, pattern-growth approaches are more flexible on the search method. Both BFS and DFS can work. Pattern-growth approaches do not need the expensive join operations used in the apriori-based approaches, nevertheless, they highly suffer the problem of duplicates generation. Indeed, the edge extension strategy can results generating the same subgraph multiple times from different extensions. Hence, existing pattern-growth approaches tried to propose ways to avoid or at least minimize the generation of duplicate subgraph candidates.

The main algorithms that have been proposed in this category are MoFa [Borgelt 2002], gSpan [Yan 2002], FFSM [Huan 2003] and GASTON [Nijssen 2004]. MoFa is mainly proposed to mine frequent subgraphs in a set of molecules. In order to accelerate the mining process, MoFa stores the embedding list of previously found subgraphs such that the extensions will be restricted only to these embeddings. Even though MoFa also uses structural and background knowledge for pruning, it still generates many duplicates.

The gSpan algorithm addresses the problem of duplication differently. It first starts generating candidates using the right-most extension technique. In this technique, according to a DFS on the graph, the path that goes straightly from the starting node to the target node is called the right-most path. Only extensions on the right-most path are allowed. It was proved that candidates generation using the right-most extension technique is complete. To alleviate the cost of isomorphism between subgraphs, gSpan uses a canonical representation where each subgraph is simply represented by a unique code called the minimum DFS code, allowing an easy detection of isomorphic subgraphs.

FFSM also uses a canonical representation, in a matrix form, called the Canonical Adjacency Matrix (CAM) to represent graphs and to detect isomorphism. It generates new subgraph candidates either by extension of a CAM or by joining two CAMs using a set of adapted operators.

In many contexts, GASTON is considered as the fastest subgraph mining

algorithm. In contrast to all existing approaches, it exploits the fact that a wide range of the discovered frequent patterns are paths and trees, and only a portion (that is sometimes very small) represents subgraphs with cycles. Hence, GASTON considers them differently by splitting the frequent subgraph mining into path mining, then subtree mining, and finally subgraph mining. Consequently, the subgraph isomorphism is only performed in the final step. GASTON also records the embedding list to save unnecessary isomorphism detection by extending only patterns that appear in the list.

### 3.4.4 Variants of frequent subgraph mining: closed and maximal subgraphs

According to the antimonotonicity property, all subgraphs of a frequent subgraph are also frequent. This arises a problem of dimensionality. Indeed, this problem becomes even more serious with large subgraphs as they contain an exponential number of smaller frequent subgraphs. To overcome this problem, variants of frequent subgraph mining have been proposed, namely closed subgraph mining and maximal subgraph mining.

**Definition 10** (*Closed subgraph*) A frequent subgraph  $g$  is said to be closed, if it has no supergraph  $g'$  ( $g \subset g'$ ) that is also frequent and has the same support.

**Definition 11** (*Maximal subgraph*) A frequent subgraph  $g$  is said to be maximal, if it has no supergraph  $g'$  ( $g \subset g'$ ) that is also frequent.

According to the definitions 10 and 11, both closed and maximal frequent subgraphs present a compact representation of the frequent subgraphs. Closed subgraph compactness is lossless since it contains all the information about the frequent subgraphs and their supports. However, maximal subgraph compactness does not consider the whole information since although all frequent subgraphs can be restored, the exact support of each subgraph is lost. The main approach that have been proposed in the literature for closed subgraph mining is CloseGraph [Yan 2003] and those for maximal subgraph mining are SPIN [Huan 2004b] and MARGIN [Thomas 2006]. Although the set of closed or maximal subgraphs is much smaller than the set of frequent ones, the number of subgraphs is still very high in real-world cases.

## 3.5 Feature selection

Feature selection is also known in the literature as pattern selection, attribute selection, variable selection or variable subset selection [Liu 1998, Guyon 2003,



[Liu 2007a, Saeys 2007, Ladha 2011]. The task of feature selection has been widely addressed in data mining not only for subgraphs but also for other types of patterns such as association rules, itemsets and sequential motifs. Although many approaches have been proposed in the literature allowing an efficient computation of frequent patterns, the number of discovered patterns is often very high. This is an obvious impact of the high dimensional nature of many types of data. Besides, most frequent pattern discovery approaches were not originally designed to consider the relevance of features.

The main goal of feature selection is to reduce the number of features by removing the *redundant* and *irrelevant* ones such that only a subset of interesting features is retained.

**Relevance of a feature:** According to an interestingness criterion, a feature is redundant if it does not bring any additional information over the currently selected features and thus it can be replaced by at least one of the already selected features. A feature is considered as irrelevant if it does not provide any useful information in any context such that it does not have any influence on the output of the prediction.

### 3.5.1 Feature selection techniques

Many approaches have been proposed for feature selection to resolve the dimensionality problem when the number of patterns is high. It is possible to categorize the existing approaches in different ways depending on the criterion used for classification. For instance, according to their relation to the learning task, feature selection approaches can be classified into:

- **Learning task dependent selection approaches** they attempt to find a subset of features that enhance the prediction capabilities of a target learning task, *i.e.*, classification, clustering, etc.
- **Learning task independent selection approaches** they tend to enhance the quality of the feature set and to remove irrelevant features without regards to the learning task.

The most conventional classification of feature selection approaches comprises three categories, namely wrapper, embedded or filter approaches [Liu 2007a].

#### 3.5.1.1 Wrapper approaches

Wrapper approaches [Cadenas 2013] are used to optimize the feature set to best fit a specific learning model. As shown in the Figure 3.4, they start

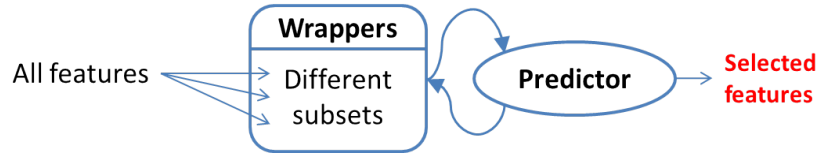


Figure 3.4: General schema of wrapper approaches for feature selection.

dividing the whole feature space into subsets using a sampling technique, generally cross-validation. Each one of the feature subsets is evaluated by training and testing the learning model. The subsets are ranked according to their predictive power and the selected subset of features is the one having the best score. Wrapper approaches are generally able to find the feature subset that best fit the learning model, but they are prone to overfitting and computationally very costly.

### 3.5.1.2 Embedded approaches

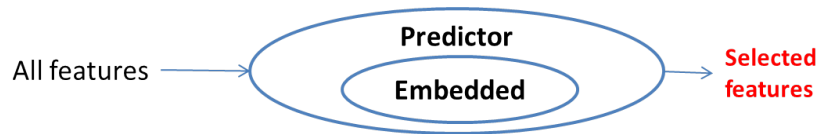


Figure 3.5: General schema of embedded approaches for feature selection.

The general schema of embedded approaches is illustrated in Figure 3.5. Similarly to wrappers, embedded approaches [Ladha 2011] are also related to a specific learning model. In contrast, the selection process is performed as part of the model construction by searching in the combined space of feature subsets. This helps alleviating the high complexity and computational cost that are in wrapper approaches. Yet, the same reason makes them also prone to overfitting.

### 3.5.1.3 Filter approaches

Filter approaches [Lazar 2012] select features independently of the chosen predictor. Figure 3.6 illustrates the general feature selection schema of filter approaches. Generally, they assess the relevance of features based on their relevance according to a significance criterion (statistical test). Based on the evaluation method of features, filter approaches can be classified into:

- **Univariate filter approaches** where features are evaluated solely according to a statistical test, such as the Pearson correlation, and without any consideration of the rest of the features.

- **Multivariate filter approaches** where features are selected based on a combinatorial search through the space of possible feature subsets. Only the most significant subset of features is selected.



Figure 3.6: General schema of filter approaches for feature selection.

As filter approaches are independent of the learning model, they can score less than wrappers and embedded approaches for a specific predictor. However, they provide a generic selection and thus they are more robust against overfitting. In addition, compared to wrappers, filter approaches are faster. Still, they are in competition with embedded approaches since the computational complexity of the latter tends to be between filters and wrappers.

### 3.5.2 Feature selection search strategy

In all the previously detailed feature selection approaches, the search in the feature space is performed following one of these two strategies:

- **Forward selection** starts with an empty set then progressively add features one by one by choosing the features that yield the best result until no more gain on the evaluation function is obtained or it reaches a specified number of features.
- **Backward selection:** starts with the whole set of features then iteratively removes them one by one. Only features with no benefits for the mining task are removed.

Since forward selection involves fewer features in the beginning, it is generally faster than backward selection. However, the latter performs better in detecting relations between features. This is because forward selection is unable to detect relations between features if one of them was not already selected in an earlier step.

## 3.6 Feature selection for subgraphs

Because of the combinatorial nature of graphs, graph mining is currently still one of the most concerned fields by the dimensionality problem. To be able to handle the exponential number of frequent subgraphs, many feature selection

approaches have been proposed for selecting significant subgraphs. Although the main goal of these approaches is to obtain a smaller yet more informative subset of subgraphs, each approach has a different way of evaluation.

### 3.6.0.1 Problem statement

In general, the process of feature selection for subgraphs can be formulated as follows. Given a graph database  $\mathcal{G} = \{G_1, \dots, G_n\}$  and an evaluation function  $F$ , find all significant subgraphs  $g^* \in \mathcal{G}$  such that :

- $g^*$  are the set of subgraphs that maximize the evaluation function  $F$ , *i.e.*,  $g^* = \operatorname{argmax}_g (F(g))$ . Or,
- $g^*$  are the subgraphs having an evaluation score that is greater or equal to a given threshold, *i.e.*,  $g^* = F(g) \geq \tau$ , if  $F$  is a threshold based function. Or,
- $g^*$  are the  $k$  subgraphs having successively the best score with  $F$ , *i.e.*,  $g^* = \operatorname{Top}_k F(g)$ .

In all three cases, the best scenario is that when the evaluation function  $F$  is embedded within the subgraph mining such that the significant subgraphs are directly mined from  $\mathcal{G}$  without needing to exhaustively generate all the set of subgraphs. However, it is not always simple to do, especially if the evaluation function is not antimonotonic. As many approaches have been (and are being) proposed for significant subgraph selection, in the following subsections, we present and discuss some of the recent and most interesting methods in the literature. We try to group them into categories based on their selection strategy.

## 3.6.1 Mining top-k subgraphs

The main idea behind top- $k$  selection approaches is that in many application domains a user may be interested in finding a specific number of patterns that best qualify to a given evaluation criteria. Thus, in this selection strategy, the methods accept a parameter  $k$  and a criterion  $F$ , then return the best  $k$  frequent subgraphs which are ranked according to  $F$ .

An interesting top- $k$  approach was proposed in [Xin 2006]. Authors proposed a greedy algorithm that approximates an optimal solution with performance bound for mining redundancy-aware top- $k$  patterns. Their algorithm was applied on graph patterns aiming to find subgraphs with the highest significance and the minimal redundancy simultaneously. The significance of a pattern  $p$  is measured using a function  $S$  such that  $S(p)$  is associated to a

real value, and  $S$  is defined by the context of application. The redundancy between pairs of patterns  $R(p, q)$  is measured by their structural similarity, *i.e.*, by computing the edit distance between them. It is worth mentioning that authors assume that the combined significance of two patterns is no less than the significance of any individual pattern and does not exceed the sum of two individual significance. Authors showed experimentally the efficiency of their approach on two real world applications namely document theme extraction and correlation-directed disk block prefetch. However, to mine the redundancy-aware top- $k$  patterns, the user needs to find all frequent patterns and assess their significance in the first place.

A recent work, termed TGP [Li 2010], mainly motivated by the fact that in most real-world cases it is difficult to select a proper value of minimum support. Indeed, if the value of the minimum support is too low thousands of patterns are extracted, but many of them are irrelevant. However, if the value of the minimum support is too high, several large subgraphs will be excluded from the extraction result. TGP is an approach for mining top- $k$  frequent closed graph patterns with size no less than a minimum size but without specifying a minimum support. It adopts a new structure called Lexicographic Pattern Net to store the patterns and relationships between them which helps facilitating the mining process. Indeed, it helps in both dynamically raising the minimum support threshold to ensure the completeness and correctness of results, and in avoiding to generate again the same candidate patterns in the next mining step. Experimental evaluation was conducted on financial, chemical compound, and synthetic datasets. Authors showed that in the case where the user is unable to provide a minimum support threshold, TGP is able to find the top- $k$  frequent closed graph patterns completely and accurately. Although TGP allows to avoid specifying a minimum support threshold, the user still has to define a proper number of patterns to select, *i.e.*, the  $k$  constraint.

### 3.6.2 Clustering-based subgraph selection

The task of clustering-based subgraph selection aims generally at obtaining a set of representative patterns, where each representative resembles a cluster centroid. In fact, clustering is the process of bringing together a set of objects into classes of similar objects. The definition of similarity between the input objects varies from one clustering model to another. In most of these models the concept of similarity is based on distances, such as Euclidean distance or cosine distance.

RP-FP and RP-GD [Liu 2007b] are two approaches for summarizing frequent graph patterns by a smaller number of representatives. Authors ex-

exploited and extended the concepts of  $\delta$ -cover and  $\delta$ -jump, to efficiently find the representative subgraphs. A graph  $g$  is  $\delta$ -covered by another graph  $g'$  if  $g \subseteq g'$  and the distance between them  $D(g, g')$  is lower than  $\delta$ . A  $\delta$ -jump subgraph  $g$  is a graph pattern such that the distance between  $g$  and any proper supergraph of  $g$  is greater than  $\delta$ . The first approach, RP-FP, selects a subset of representative patterns from a set of closed frequent subgraphs. The RP-FP method works well when the size of the set of frequent closed subgraphs is not very large. However, in real applications where the number of frequent closed patterns is usually high, RP-FP does not scale well. Therefore, authors proposed the second approach RP-GD which directly mines representative graph patterns from graph databases with lower tightness on the summarization quality. RP-GD calculates the representative set of patterns simultaneously during the extraction of frequent closed patterns. Thus, when the number of closed graph patterns is very large, RP-GD is much more efficient than RP-FP. Experiments conducted by authors on chemical compound and synthetic graph databases showed that RP-GD is much more efficient than RP-FP while achieving comparable summarization quality. Yet, the user has to define the right value of  $\delta$ .

RING [Zhang 2009] is another clustering-based subgraph selection approach. It aims at finding a subset of  $N$  representative subgraphs among the frequent ones. The representative subgraphs should satisfy the distinction where all representative subgraphs should be dissimilar to each other, and the completeness where the number of frequent subgraph patterns which cannot be delegated by the representative patterns should be as small as possible. RING adopts the idea of transforming each subgraph into a vector of invariants. Then, it uses the euclidean distance between these vectors to compute the distance between subgraphs instead of using the costly edit distance measure (the minimum amount of deletion/insertion of nodes and edges to transform one graph into another). To find the  $N$  representative subgraphs, all frequent subgraphs are grouped into  $N$  clusters, and the subgraphs that are closest to the clusters centers are selected as representatives. Authors also proposed a way of integrating RING selection into the mining process of frequent subgraphs to directly discover the representative subgraphs. They process the subgraph discovery in a DFS manner. For any frequent subgraph  $P_1$  represented by another representative subgraph  $P_2$ . If while  $P_1$  is being extended another pattern  $P_{11}$  represented by the same representative  $P_2$  has been reached then the growing stops. If such subgraph has not been reached, the growing continue and if the frequent subgraph is not covered by any one of the representatives, then it is considered as a new representative. The DFS mining algorithm stops when all the supergraphs of existing representative are not frequent. RING was tested on a synthetic graph dataset as well as on a

graph representing a protein-protein interaction network. RING was able to select representative subgraphs in a fast way. Yet, its pruning condition assumes that the supergraphs of the pruned subgraph will be also represented by the same representative which is not always true. This may prevent reaching many representatives and thus leading to poor selection.

### 3.6.3 Sampling-based approaches

In statistics, sampling can simply be defined by the task of selecting a subset of individuals among a given statistical population to estimate the characteristics of the whole set. In our context (*i.e.*, pattern selection), sampling is a method for selecting a subset of  $n$  patterns out of  $N$  such that the sampled  $n$  patterns allows (or approximately) estimating the characteristics of all the  $N$  patterns. Sampling-based approaches are mainly proposed due to the assumption that in many real application contexts, it is very costly or even impossible to generate the entire set of frequent patterns. Thus, sampling-based selection approaches tries to approximately generate the set of significant patterns by only considering a sample of the entire set of patterns.

ORIGAMI [Hasan 2007] is a method for the extraction of representative subgraphs. Unlike most of the existing selection approaches that considers relations between patterns in the transaction space to obtain representatives, ORIGAMI considers the distances in the pattern space which is obviously more complex especially in the case of graph patterns. ORIGAMI is composed of two steps. First, it starts by extracting a sample of frequent maximal subgraphs through a random walk along the frequent subgraph lattice. Then straightforwardly, it selects a subset of  $\alpha$ -orthogonal (non-redundant) and  $\beta$ -representative subgraphs. Two subgraphs are  $\alpha$ -orthogonal if their similarity is bounded above by  $\alpha$ , and a subgraph is said to be  $\beta$ -representative for another if their similarity is at least  $\beta$ . The similarity between two subgraphs is computed by finding how much their maximal common subgraph [Abu-Khzam 2007] represent from their overall structure. Experiments were conducted on various types of datasets: chemical compounds, protein structures, protein-protein interaction network and a synthetic dataset. They showed the efficiency and scalability of the approach. However, the selection in ORIGAMI is straightforward and is performed after discovering the frequent maximal subgraphs. In addition, the randomized search used in ORIGAMI risks walking only over a portion of the frequent subgraph space while the rest of it is ignored. This may leads to discover poor quality representatives.

In [Hasan 2009], authors proposed an output space sampling approach for subgraphs. It samples interesting subgraph patterns without enumerating the entire set of candidate frequent patterns. The sampling is driven by a distri-

bution that is predefined by the user. This is performed through a random walk on the candidate subgraph partial order. When the walk converges to a desired distribution, the algorithm stops and returns the discovered subgraph samples. Experimental analysis was performed on graph datasets (namely chemical compound, protein interaction and cell-graphs datasets) as well as on an itemset dataset. Results showed that the output space sampling approach is scalable and able to discover a sample of significant patterns according to the desired distribution. This approach is useful in the case where traditional approaches fail to run. Although authors successfully performed experiments on small and large graph datasets, their approach stores the entire database in the memory which makes it inefficient if the database does not fit the memory. Besides, the user should carefully define the distribution parameter since it highly affects the quality of the sampling.

In [Schietgat 2011], authors proposed an approach for mining a specific type of subgraph patterns. The main idea of their approach is to mine the maximum common subgraphs (MCSs) [Abu-Khzam 2007] between pairs of graphs in the database. The pairs of graphs are randomly selected from the database and thus the mined subgraphs represent only a sample from the entire set (MCSs sample). Experimental evaluation was performed on 60 benchmark datasets of molecular compounds representing problems from chemoinformatics. To evaluate the quality of the mined subgraphs, they were used as features for classification. Results show that their approach produces a smaller and less redundant set of subgraph patterns, and allows better prediction performance in classification than other state-of-the-art approaches. Besides, it is parameter free and runs in polynomial time. However, the proposed approach is restricted to outerplanar graphs [Schietgat 2011]. Although it works well with chemical compounds datasets, this approach may not scale well with larger graphs (such as the case of graphs representing protein 3D-structures), and larger datasets since the search is pairwise.

### 3.6.4 Approximate subgraph mining

Approximation is usually used when the exact result is unknown or difficult to obtain such that the obtained inexact result are within required limits of accuracy. In many applications, mining the whole set of frequent subgraphs is very difficult. Besides, in applications like the analysis of protein interaction networks and social networks, slight differences between subgraphs may not be important. In such cases, approximation is a way to handle both issues by only enumerating an approximate set of subgraphs such that similar but slightly different subgraphs will collapse into one representative. Approximation in subgraph mining is performed by structural approximation or label



approximation.

An interesting work that falls in this category is [Chen 2008], where authors proposed an approach for mining a set of structural representative subgraphs among the whole set of frequent ones. After mining all frequent subgraphs, they perform a *smoothing-clustering* selection that is based on approximate structural similarity on micro and macro sides. In the smoothing step, they consider a tolerance threshold to summarize approximately isomorphic subgraphs into one representative. In the clustering step, they collapse multiple structurally similar subgraphs into one representative using a clustering algorithm where the center of the cluster is considered to be the representative subgraph delegate. They used K-Medoids [Ng 2002] for non-overlapping clustering and  $\varepsilon$ -bounded clustering [Hochbaum 1997] for overlapping clustering where each subgraph may belong to multiple clusters and thus may be represented by more than one structural representative. Efficiency of the *smoothing-clustering* approach was evaluated through experiments on chemical compounds and synthetic graph dataset. Yet, in order to find the representatives, all frequent subgraphs have to be discovered first. Besides, the method contains many parameters that need to be properly defined in order to prevent poor selection.

Unlike structural approximation, less attention have been devoted to label approximation. A very recent work have been developed in [Anchuri 2013] for mining significant subgraphs based on label approximation. This work operates in the context of single large graph, however, we are here interested in approaches devoted to graph databases. Thus, this approach is more discussed in chapter 4 as a matter of comparison. Structural approximation is also more discussed in chapter 5.

### 3.6.5 Discriminative subgraph selection

Supervised classification is one of the important applications that use frequent subgraphs and in general frequent patterns. In the case where graphs in the database are labeled (*i.e.*, each graph is affiliated to a class), it is possible to extract discriminative frequent subgraphs. These subgraphs are lately used as attributes for machine learning classifiers to help building models that best discriminate between classes. Several selection methods and algorithms have been proposed for mining discriminative frequent graph patterns. In the following, we list and explain some of the best known in the literature.

### 3.6.5.1 Filter methods

In [Yan 2008], authors proposed LEAP, an approach for mining the most discriminative subgraphs. LEAP is designed to exploit the correlation between patterns through both structural similarity and significance. It iteratively looks for the optimal patterns until all graphs are covered. LEAP was tested on a collection of molecular compound datasets transformed into graphs. It is able to quickly locate few number of highly discriminative subgraphs without exploring the whole pattern space. The selected subgraphs facilitate the training of a classification model and help enhancing its accuracy.

GraphSig [Ranu 2009] provides a solution to mining discriminative subgraph patterns with low frequencies. It starts by converting graphs into feature vectors through a random walk with restarts on each node. Domain knowledge is used to select a meaningful feature set. GraphSig assumes that graphs with similar feature vectors share highly frequent subgraphs. Thus, it clusters the graphs having similar feature vectors into small groups. After that, it mines frequent subgraphs in each group with high frequency thresholds ensuing a high reduction in the computation cost. GraphSig was tested on molecular compound datasets. Results show that GraphSig is scalable and able to find discriminative patterns in large graph datasets and even with low frequencies.

GAIA [Jin 2010] is another approach for mining discriminative subgraphs for graph classification. GAIA adopts evolutionary computation in discriminative subgraph mining through a randomized exploration of the candidate subgraphs search space. Further, as the search is randomized, GAIA uses parallel computation to improve the quality of the set of selected discriminative patterns by integrating the results from independent instances of pattern evolution. After discovering the discriminative patterns, GAIA employs sequential coverage and uses the mined patterns to generate association rules as graph classifiers. Experimental evaluation of GAIA was performed on a number of protein and chemical compound datasets. It showed that due its parallelization technique, GAIA is scalable and able to quickly find highly discriminative subgraphs.

CORK [Thoma 2010] is a subgraph selection method for mining discriminative subgraphs. The main idea in CORK is to preserve subgraphs that eliminate the correspondence between graphs of different classes and that also enhance the discrimination power of the set of already selected subgraphs. It attempts to discover frequent subgraphs that are most discriminative for classification using a submodular quality function. Authors showed that the used submodular quality function criterion can be integrated into the state-of-the-art tool for frequent subgraph mining gSpan, allowing fast pruning of

the search space for discriminative frequent subgraphs mining. Efficiency of CORK was evaluated on various protein and chemical compound datasets. Results showed that CORK works well with two classes as well as with multi-class classification problems.

In [Zhu 2012], authors proposed an approach for mining diversified discriminative subgraphs termed D&D. The main idea in D&D is that besides a discrimination measure it additionally explores the diversity between subgraphs. Diversity is considered by reducing the overlapping between a new candidate subgraph and the already selected subgraphs, based on an edge-cover. To further enhance diversity, it also considers how to reduce the overlapping between the occurrence list of the candidate subgraph and those of all the already selected subgraphs. Experimental analysis on protein and chemical compound datasets showed that besides the discriminative power, considering the diversity between subgraphs during the selection highly affects the results and enhances the classification by making the positive and negative graphs more separable.

### 3.6.5.2 Boosting methods

In the previously mentioned selection methods, the training and building of classification models are performed separately, *i.e.*, after mining the set of discriminative features. There exists other methods in the literature where the search for discriminative subgraphs is performed at the same time as the construction of the classification models. In the following, we try to cover some of the most interesting works in the literature.

An approach termed gPLS was proposed in [Saigo 2008]. It uses DFS to mine the frequent graph patterns. It adapts the powerful mathematical tool of PLS (Partial Least Squares) regression to graph mining to select informative subgraphs then uses them to directly build a classifier with fewer iterations than typical boosting methods. The gPLS algorithm was evaluated on chemical compound datasets. It creates latent variables involving response variables, thus leading to better predictions. However, these latent variables have the known disadvantage of poor interpretability.

COM [Jin 2009] is a graph classification method which follows a process of pattern mining and classifier learning. COM employs a pattern exploration order such that the complementary discriminative patterns are examined first. Based on the subgraphs co-occurrences information, it constructs classification rules by assembling weak features in order to generate strong ones. Patterns are grouped into co-occurrence rules during the pattern exploration, leading to an integrated process of pattern mining and classifier learning. Evaluation of COM on protein and chemical compound datasets showed that it has com-

petitive results in terms of classification accuracy and execution time. Besides, it produces an interpretable classifier.

The gBoost algorithm [Saigo 2009] is a mathematical programming boosting method for classifying labeled graphs. It progressively collects discriminative subgraph patterns through a branch-and-bound pattern search algorithm based on the DFS code tree. The search algorithm of gBoost is integrated into gSpan. It uses the class labels as an extra information source for pruning the search space and also reuse the constructed search space in later iterations to minimize the computation time. gBoost repeatedly constructs multiple weak classifiers where each weak classifier (called decision stump) uses a subgraph as a classification feature. Experiments of gBoost were conducted on chemical compound datasets. They showed that gBoost scores very high in classification as well as in regression. In addition, it is flexible and can be coupled with any pattern mining algorithm.

In [Fei 2010], authors designed LPGBCMP, a boosting method for graph classification. It selects clustered features by considering the structure relationship between subgraph patterns in the functional space. The selected subgraphs are used as weak classifiers (also called base learners) to obtain high quality classification models. Authors theoretically proved that LPGBCMP exhibits a natural grouping effect for nearby spatial or overlapping features, and they showed that the proposed method can be naturally extended to other semi-structured data such as sequences. Experimentally, they evaluated LPGBCMP on classifying a set of protein datasets. Results showed that their approach provides high classification performance.

A common drawback of all these methods is that in the case where graphs in the database has no class labels, they become useless. Besides, they considers the discrimination power of patterns individually. This may fail if no individual pattern has high discrimination power, yet, jointly some patterns may have higher discrimination. Moreover, the selected patterns may be individually discriminative but redundant if there exists a significant overlap in their supporting graphs. This makes them more vulnerable to overfitting.

### 3.6.6 Other significant subgraph selection approaches

Some other selection approaches have been proposed for mining significant subgraphs, not necessarily for a specific application context, but only for extracting relevant or hidden information or to simply characterize a given dataset. Many of these approaches have their own original and unique selection technique and thus it is difficult to classify all of them under one of the above subgraph-selection categories. In the following, we present some of the most interesting approaches.

In [Pennerath 2009], authors were interested in mining a small number of patterns for characterizing a given dataset. The selected patterns should guarantee maximum informativeness and minimum redundancy. They introduced the class of Most Informative Patterns (MIPs). In fact, structural redundancy is assessed in MIPs through mining only the closed patterns and not all the frequent ones, and the informativeness of a pattern is measured with respect to a scoring function given by the expert. Authors presented two algorithms for extracting MIPs: the first one directly searches for MIPs in a dataset while the second one selects MIPs from frequent patterns. They showed that MIPs can be used on different kinds of patterns and they applied it on itemsets and subgraphs. Experimental analysis was performed on different itemset datasets and a chemical compound graph dataset. Results showed that MIPs is able to provide a reduced set of patterns that are representative of a dataset. Yet, the selected MIPs still contains some redundancy. In addition, the redundancy function is user-defined. Although this makes the approach flexible, it may be a hard task in some applications.

Recently, some approaches for patterns selection have adopted the notion of dominance between patterns to mine *skyline* patterns [Papadopoulos 2008, Soulet 2011, Bouker 2012]. In [Papadopoulos 2008], authors proposed SkyGraph a selection approach dedicated to graph patterns. The main goal of this approach is to consider simultaneously, in the selection, a set of user-defined criteria. These criteria are usually in the form of interestingness measures. The set of defined measures are considered together through skyline processing. In skyline processing, the patterns returned to the user are the ones that are not dominated by any other pattern. A pattern  $P_1$  dominates another pattern  $P_2$  if  $P_1$  is as good as  $P_2$  with all measures and  $P_1$  scores better than  $P_2$  in at least one measure. The higher the subgraph scores with the measures, the more important it becomes. SkyGraph was applied in the context of graphs towards retrieving skyline subgraphs. Experiments were conducted on graph databases representing a microarray network, road network of San Francisco, and co-authors network using two measures to determine the importance of subgraphs namely the order of the subgraph (the number of nodes) and the subgraph edge connectivity. They showed that the dominance relation allows SkyGraph to detect important subgraphs. However, SkyGraph risks selecting only one subgraph in the case where the latter outperforms all the other subgraphs for the considered measures. Thus, analyzing the correlation between the considered measures may help avoiding such problem. Besides, introducing a ranking function to score the patterns may help specifying a sufficient number of patterns in the case where the selected ones is not enough for the application.

Some other works addressed the interestingness of patterns differently,

they assessed the significance of patterns based on their structure. Depending on the application, it may be interesting to target patterns having a specific type of structure. An interesting example of such approaches was proposed in [Saidi 2012] where authors proposed an approach for extracting the so-called *ant-motifs* from protein 3D-structures. First, they transform protein 3D-structures into protein graphs, then they try to discover common substructures having an ant-like shape such that each substructure is mainly composed of a fragment from the primary structure that is enriched with other distant amino acids that are directly linked to the considered fragment. Experimental evaluation was conducted on real protein graph datasets for classification. They showed that ant-motifs outperform frequent subgraphs in classification. Besides, ant-motifs are based on biological basis and their number is significantly smaller than that of frequent subgraphs. Although theoretically the approach can be used with traceable graphs (having Hamiltonian paths), it is still currently limited to protein graphs and no other application contexts have been tested so far.

### 3.7 Discussion

As there exists currently many subgraph selection approaches, it is difficult and even unfair to compare them in general since the majority of the approaches were originally designed to solve a particular issue. Hence, the choice of an appropriate selection method highly depends on the users' needs and the application constraints. In order to help assisting such choice, in Table 3.1, we list all the subgraph selection approaches that have been investigated in this chapter and we state their characteristics according to a set of descriptors.

Table 3.1: Characteristics of Subgraph selection approaches according to different discription criterions.

<i>Subgraph selection approach</i>	<i>Descriptor</i>		
	Post-processing	Learning-task dependent	Selected subgraphs
TGP	No	No	Top-k frequent closed
Redundancy aware top-k	Yes	No	Top-k frequent significant&non-redundant
RP-FP	Yes	No	Frequent closed representatives
RP-GD	No	No	Frequent closed representatives
RING	No	No	Frequent representatives
ORIGAMI	Yes	No	$\alpha$ -orthogonal $\beta$ -representative
Output space sampling	No	No	Sample of frequent
MCSs sample	No	No	Maximum common subgraphs
Smoothing-clustering	Yes	No	Approximate structural representatives
D&D	No	Yes	Diverse discriminative
GAIA, CORK, GraphSig, LEAP, LPGBCMP, COM, gBoost, gPLS	No	Yes	Discriminative
MIPs	No	No	Most informative closed
SkyGraph	No	No	Undominated
Ant-motifs	No	No	Ant-like shape

## 3.8 Conclusion

Frequent subgraph discovery is one of the most important mining techniques in graph mining. Because of the high number of frequent subgraphs, many subgraph selection approaches have been proposed in the literature. In general, they attempt to resolve the dimensionality problem by assessing the redundancy or the relevance of subgraphs through similarity or interestingness measures. In this chapter, we first presented the context and formalization of frequent subgraph discovery and the main approaches and techniques proposed in the literature. We also presented the general framework for feature selection and the complexity of adopting the old techniques for frequent subgraphs. We further investigated a panoply of the most interesting subgraph selection approaches proposed in the literature. In the next two chapters, we propose and discuss two novel selection approaches for subgraph patterns namely UnSubPatt and TRS.





# Part II

## Contributions



# UnSubPatt: Mining representative unsubstituted graph patterns by means of substitution matrices

---

## Contents

<b>4.1</b>	<b>Aims</b>	<b>62</b>
<b>4.2</b>	<b>Introduction</b>	<b>62</b>
<b>4.3</b>	<b>Mining representative unsubstituted patterns</b>	<b>65</b>
4.3.1	Background	65
4.3.2	Preliminaries	67
4.3.3	Algorithm	70
4.3.4	Illustrative example	72
<b>4.4</b>	<b>Experiments</b>	<b>73</b>
4.4.1	Datasets	73
4.4.2	Protocol and settings	75
<b>4.5</b>	<b>Results and discussion</b>	<b>75</b>
4.5.1	Empirical results	76
4.5.2	Results using other substitution matrices	77
4.5.3	Impact of varying the substitution threshold	77
4.5.4	Smoothing the distribution of patterns	81
4.5.5	Comparison with other approaches	81
4.5.6	Runtime analysis	84
<b>4.6</b>	<b>Conclusion</b>	<b>85</b>

---

## 4.1 Aims

In the previous two chapters, we investigated methods for transforming protein structures into protein graphs enabling using graph mining techniques to study them. We also mentioned that mining frequent subgraphs is one of the best ways to analyze graph data, yet, it suffers from the curse of dimensionality. We showed how to overcome this issue through feature selection, and we reviewed methods from the literature for subgraph selection. In existing subgraph selection approaches, the *prior* information and knowledge about the application domain are often ignored. However, the latter provide valuable knowledge that may help building dedicated approaches that best fit the studied data. In our context, the existence of substitution matrices for the amino acids composing protein structures, represents a valuable domain knowledge that can be exploited. In this chapter, we propose a novel feature selection approach for subgraphs. It selects a subset of so-called *representative unsubstituted* subgraphs from the frequent ones by incorporating a specific domain knowledge which, in our context, consists of the protein substitution matrices.

## 4.2 Introduction

Studying protein structures can reveal relevant structural and functional information which may not be derived from protein sequences alone. During recent years, various methods that study protein structures have been elaborated based on diverse types of descriptors such as profiles [von Öhsen 2004], spatial motifs [Kleywegt 1999, Sun 2012] and others [Mavridis 2010]. Besides, the exponential growth of online databases such as the Protein Data Bank (PDB) [Berman 2000], CATH [Cuff 2011], SCOP [Andreeva 2008] and others, arises an urgent need for more accurate methods that will help to better understand the studied phenomena such as protein evolution, functions, etc.

In this scope, proteins have recently been interpreted as graphs of amino acids and studied based on graph theory concepts [Vishveshwara 2002, Huan 2004a]. This representation enables the use of graph mining techniques to study protein structures in a graph perspective. In fact, in graph mining, any problem or object under consideration is represented in the form of nodes and edges and studied based on graph theory concepts [Bartoli 2007, Hasan 2009, Jin 2009, Cheng 2010]. As mentioned in the previous chapter, one of the powerful and current trends in graph mining is frequent subgraph discovery. It aims to discover subgraphs that frequently occur in a graph dataset and use them as patterns to describe the data. These patterns are lately analyzed by domain experts to reveal interesting informa-

tion hidden in the original graphs, such as discovering pathways in metabolic networks [Faust 2010], identifying residues that play the role of hubs in the protein and stabilize its structure [Vallabhajosyula 2009], etc.

The graph isomorphism test is one of the main bottlenecks of frequent subgraph mining. Yet, many algorithms have been proposed in the literature and made it feasible for instance FFSM [Huan 2003], gSpan [Yan 2002], GASTON [Nijssen 2004], etc. Unfortunately, the exponential number of discovered frequent subgraphs is another serious issue that still needs more attention, since it may hinder or even make any further analysis unfeasible due to time, resources, and computational limitations. This problem becomes even more serious with graphs of higher density such as those representing protein structures. In fact, the issues raised from the huge number of frequent subgraphs are mainly due to two factors, namely *redundancy* and *significance* [Thoma 2010]. Redundancy in a frequent subgraph set is caused by structural and/or semantic similarity, since most discovered subgraphs differ slightly in structure and may infer similar or even the same meaning. Moreover, the significance of the discovered frequent subgraphs is only related to frequency. This yields an urgent need for efficient approaches allowing to select relevant patterns among the large set of frequent subgraphs.

In this chapter, we propose a novel selection approach which selects a subset of representative patterns from a set of labeled subgraphs, we term them *unsubstituted patterns*. In order to select these unsubstituted patterns and to shrink the large size of the initial set of frequent subgraphs, we exploit a specific domain knowledge, which is the substitution between amino acids represented as nodes. The main contribution of this work is to define a new approach for mining a representative summary of the set of frequent subgraphs by considering the ability of substitution between nodes' labels of the graph which is defined in the domain knowledge. In this work, we apply the proposed approach on protein structures because of the availability of substitution matrices in the literature. However, it can be considered as general framework for other applications whenever it is possible to define a matrix quantifying the possible substitutions between the labels. For instance, in graphs representing protein-protein interaction networks, each node of the graph represent a protein in the network and these proteins share structural and sequential similarities. Since it is possible to measure such similarity using for instance an alignment tool, then it would be possible to define a matrix quantifying similarities between the labels. Another possible application example is ontology alignment. Ontology alignment refer to the process of determining correspondences between concepts. Each ontology can be represented in the form of a graph were the nodes represent the concepts. Since the concepts are semantically similar, it is possible to define a matrix quantifying these simi-

larities and thus to adapt our approach to be used for detecting similarities between ontologies. Our approach can also be used on any type of subgraph structure such as cliques, trees and paths (sequences). In addition, it can be easily coupled with other pattern selection methods such as discrimination or orthogonality based approaches. Moreover, unlike other approaches that are supervised and learning task dependent, this approach is unsupervised and can help in various mining tasks.

Recently, several approaches have been proposed for pattern selection in subgraph mining. To the best of our knowledge, in all existing subgraph selection approaches, the selection is usually based on structural similarity [Hasan 2007] and/or statistical measures (e.g. frequency and coverage (closed [Yan 2003], maximal [Thomas 2006])), discrimination [Thoma 2010], *etc*). Yet, the *prior* information and knowledge about the application domain are often ignored. However, these prior knowledge may help building dedicated approaches that best fit the studied data.

A very fresh work have been proposed in [Anchuri 2013], where the authors presented an approach for mining an approximate set of frequent subgraph patterns from a single large graph database in the presence of a cost matrix label. This approach is very similar to the one we propose in this chapter, in the sense that both approaches aim to find representative subgraphs, and incorporate a matrix that defines distances (*i.e.*, similarities) between the labels. In addition, both approaches preserve the topology of subgraphs but allow bounded label mismatches. However, they have many differences. First of all, their approach is used in the context of mining representative subgraphs from a single large graph, however our approach is used with graph databases. In addition, both approaches differ in the way of exploiting the label matrix, *i.e.*, in the way they measure similarities between subgraphs. Moreover, their approach is based on sampling and thus it generates an approximate set of representative subgraphs, but our approach ensures generating the optimal set of representatives. A major advantage of their approach is that it is incorporated in the extraction process of frequent representative subgraphs, while our approach operates in post-processing. Although this makes their approach more appropriate to very large graphs, our approach is more efficient in the case of moderate and small graph databases.

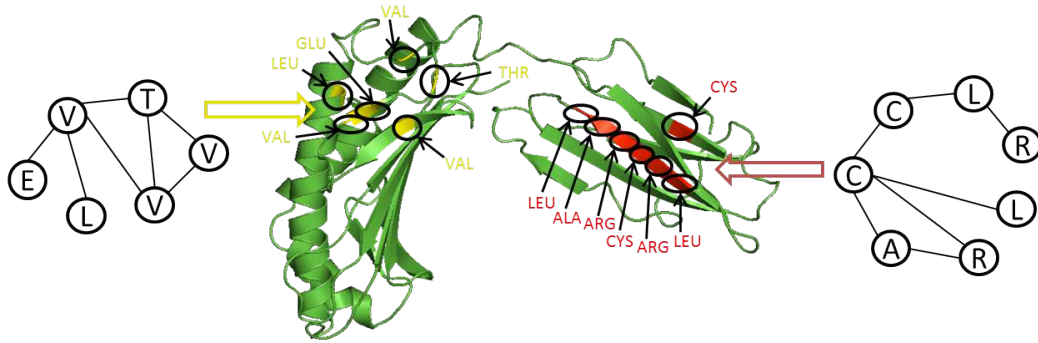


Figure 4.1: An example of two representative unsubstituted patterns sampled from our experiments (from C-type lectin domains (DS3)). The figure also shows the corresponding position of each node of the subgraphs in the real protein 3D-structure.

## 4.3 Mining representative unsubstituted patterns

### 4.3.1 Background

Statistical pattern selection methods have been widely used to resolve the dimensionality problem when the number of discovered patterns is too large. However, these methods are too generic and do not consider the specificity of the application domain and the used data. We believe that in many contexts, it would be important to incorporate the background knowledge about the application domain in order to create approaches that best fit the considered data. In proteomics, a protein structure is composed by the folding of a set of amino acids. During evolution, amino acids can substitute each other. The scores of substitution between pairs of amino acids were quantified by biologists in the literature in the form of substitution matrices such as Blosum62 [Eddy 2004]. Our approach uses the substitution information given in the substitution matrices in order to select a subset of unsubstituted patterns that summarizes the whole set of frequent subgraphs. We consider the selected patterns as the representative ones. Figure 4.1 shows two representative unsubstituted patterns (sampled from DS3 (C-type lectin domains) from our experiments) and shows the corresponding position of each node of the subgraphs in the real protein 3D-structure.

The main idea of our approach is based on node substitution. Since the nodes of a protein graph represent its amino acids, hence, using a substitution matrix, it would be possible to quantify the substitution between two given subgraphs. Starting from this idea, we define a similarity function that mea-



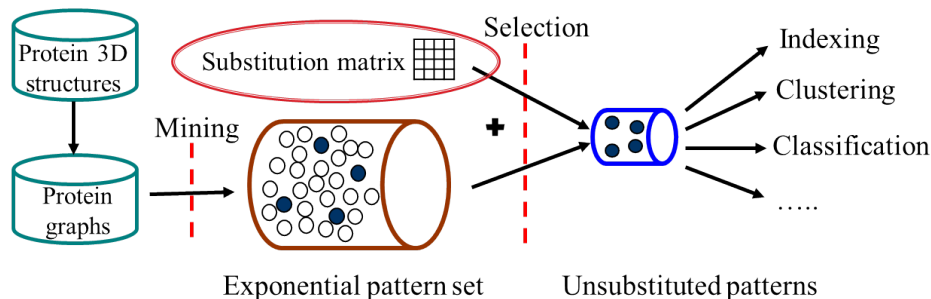


Figure 4.2: Unsubstituted pattern selection framework.

sures the distance between pairs of subgraphs. Then, we preserve only one subgraph from each similar pair with respect to a user-specified threshold such that the preserved subgraphs represent the set of representative unsubstituted patterns. An overview of the proposed approach is illustrated in Figure 4.2 and a more detailed description is given in the following sections.

The substitution between amino acids was also used in the literature but for sequential feature extraction from protein sequences in [Saidi 2010], where the authors proposed a novel feature extraction approach termed *DDSM* for protein sequence classification. Their approach is restricted to protein sequences and generates every subsequence substituting another one. In other words, DDSM eliminates any pattern substituted by another one and which itself does not substitute any other one. We believe that their approach does not guarantee an optimal summarization since its output may still contain patterns that substitute each other. In addition, they do consider negative substitution scores as impossible substitutions which is biologically not true since negative scores are only expressing the less likely substitutions, which obviously does not mean that they are impossible. Moreover, DDSM is limited to protein sequences and does not handle more complex structures such as the protein 3D-structure. Our approach overcomes these shortcomings, since it can handle both protein sequences and 3D-structures (since a sequence can be seen as a line graph). In addition, it considers both the positive and negative scores of the substitution matrix. Moreover, our approach generates a set of representative unsubstituted patterns ensuring an optimal summarization of the initial set. Besides, it is unsupervised and can be used in classification as well as in other analysis and knowledge discovery contexts unlike DDSM which is dedicated to classification.

### 4.3.2 Preliminaries

Here, we present the preliminaries and the formal statement of the proposed approach. Let  $\mathcal{G}$  be a dataset of protein structures represented as graphs. Each graph  $G = (V, E, \Sigma, L)$  of  $\mathcal{G}$  is given as a collection of nodes (amino acids)  $V$  and edges (interactions)  $E$ . The nodes of  $V$  are labeled within an alphabet  $\Sigma$  (amino acids types) and  $L$  is the label function that maps each node in  $V$  to a label in  $\Sigma$ . We denote by  $|V|$  the number of nodes (the graph order) and by  $|E|$  the number of edges (the graph size). Let also  $\Omega$  be the set of frequent subgraphs extracted from  $\mathcal{G}$ , also referred here as *patterns*.

**Definition 12** (*Substitution matrix*) Let  $\mathcal{A}$  be a substitution matrix defined over  $\Sigma$ .  $\mathcal{A}$  is defined as follows:

$$\mathcal{A} : \begin{cases} \Sigma^2 & \longrightarrow [\perp, \top] \subset \mathbb{R} \\ (l, l') & \longrightarrow x \end{cases} \quad (4.1)$$

where  $l, l' \in \Sigma$  and  $x$  is the substitution score between the labels  $l$  and  $l'$ .

The higher the value of  $x$  is, the more likely is the substitution of  $l'$  by  $l$ . If  $x = \perp$  then the substitution is impossible, and if  $x = \top$  then it is certain, i.e., the substitution should happen. The values  $\perp$  and  $\top$  are optional and user-specified. They may appear or not in  $\mathcal{A}$ . Table 2.2 shows a real example of a protein substitution matrix defined over the amino acids types.

In proteins' substitution matrices, both positive and negative values represent possible substitutions. However, positive scores are given to the more likely substitutions while negative scores are given to the less likely ones. In order to give more magnitude to higher values of  $x$ , we define  $\mathcal{M}$  over  $\Sigma$  such that  $\forall l, l' \in \Sigma$ :  $\mathcal{M}(l, l') = e^{\mathcal{A}(l, l')}$ . As we consider only substitutions between patterns having the same structure, we define the structural isomorphism as follow:

**Definition 13** (*Structural isomorphism*) Two patterns  $P = (V_P, E_P, \Sigma, L)$  and  $P' = (V_{P'}, E_{P'}, \Sigma, L)$  are said to be structurally isomorphic (having the same shape), we note  $\text{shape}(P, P') = \text{true}$ , iff:

- $P$  and  $P'$  have the same order and size, i.e.,  $|V_P| = |V_{P'}|$  and  $|E_P| = |E_{P'}|$ ,
- $\exists$  a bijective function  $f : V_P \rightarrow V_{P'} : \forall u, v \in V_P$ , if  $\{u, v\} \in E_P$  then  $\{f(u), f(v)\} \in E_{P'}$  and inversely.

It is worth mentioning that in this definition, we consider only the isomorphism on structure and we ignore the labels.

As we are seeking the most representative patterns based on the substitution of amino acids, the best representatives are supposed to be the ones that represent as much other patterns as possible. Thus, each representative should have the highest probability of mutation to all the patterns it substitutes, *i.e.* the **most mutable one**. It is also possible to choose the least mutable pattern over the most mutable one, however, a good representative pattern is supposed to be the one having the maximal overall similarity to all the other patterns it represents. Patterns with higher ability of mutation are supposed to substitute more other patterns which allows a better summarization of the pattern set. Based on these assumptions, we are considering the most mutable patterns as the representatives. We define the pattern mutation score as follows:

**Definition 14** (*Pattern mutation score*) Given a pattern  $P = (V_P, E_P, \Sigma, L) \in \Omega$ , the pattern mutation score  $M_{patt}(P)$ , measures the possibility that  $P$  mutates to any other pattern having the same order.

$$M_{patt}(P) = 1 - \prod_{i=1}^{|V_P|} M_{el}(V_P[i]) \quad (4.2)$$

where  $\prod_{i=1}^{|V_P|} M_{el}(V_P[i])$  represents the score that the pattern  $P$  **does not** mutate to any other pattern (*i.e.*  $P$  stays itself), and  $M_{el}(V_P[i])$  represents the elementary conservation score for each node  $V_P[i] \in V_P$ . Precisely, given a node  $v$  having a label (amino acid type)  $l \in \Sigma$ ,  $M_{el}(v)$  measures the possibility that  $v$  **does not** mutate to any other node depending on its label  $l$ :

$$M_{el}(v) = \frac{\mathcal{M}(l, l)}{\sum_{i=1}^{|\Sigma|} \mathcal{M}(l, l_i)} \quad (4.3)$$

The lower the values of elementary conservation of nodes are, the more is the mutation ability of the pattern.

Based on the pattern mutation score, we are able to rank patterns and thus to chose between each substitutable pair of patterns which one is supposed to be the representative. We define the pattern substitution score which measures the possibility that a pattern substitutes another one.

**Definition 15** (*Pattern substitution score*) Given two structurally isomorphic patterns  $P$  and  $P'$ , we denote by  $S_{patt}(P, P')$  the substitution score of  $P'$  by  $P$ . It measures the possibility that  $P$  mutates to  $P'$ . Formally:

$$S_{patt}(P, P') = \frac{\sum_{i=1}^{|V_P|} S_{el}(V_P[i], V_{P'}[i])}{|V_P|} \quad (4.4)$$

$S_{el}(V_P[i], V_{P'}[i])$  measures the possibility that the node  $V_P[i]$  substitutes the node  $V_{P'}[i]$  based on the substitution scores between their amino acids types. Obviously and according to all proteins' substitution matrices, for any amino acid  $l$  there is only another one that best substitutes it. It is obviously itself. Accordingly, given two nodes  $v$  and  $v'$  having correspondingly the labels  $l, l' \in L$ , the elementary substitution score between  $v$  and  $v'$ , denoted by  $S_{el}(v, v')$ , is computed as follows:

$$S_{el}(v, v') = \frac{\mathcal{M}(l, l')}{\mathcal{M}(l, l)} \quad (4.5)$$

**Definition 16** (*Pattern substitution*) Based on definitions 13 and 15, we say that a pattern  $P$  substitutes  $P'$ , we note  $\text{subst}(P, P', \tau) = \text{true}$ , iff:

1.  $P$  and  $P'$  are structurally isomorphic (i.e.,  $\text{shape}(P, P') = \text{true}$ ),
2. The score of substitution of  $P'$  by  $P$  is greater than a given user-threshold  $\tau$  (i.e.,  $S_{\text{patt}}(P, P') \geq \tau$ ), where  $0\% \leq \tau \leq 100\%$ .

Since we are proposing a pattern selection approach, the output set of representative patterns should be as small as possible. It is not supposed to have any pair of substitutable patterns such that it contains only the representative ones which guarantees a maximal summarization.

**Definition 17** (*Unsubstituted pattern*)  $\Omega^* \subset \Omega$  represents the subset of representative unsubstituted patterns if and only if there does not exist any pair of patterns  $(P_1, P_2)$  in  $\Omega^*$  that are substitutable, with respect to the minimum substitution threshold  $\tau$ .

A pattern  $P^*$  is considered as representative unsubstituted pattern, i.e.,  $P^*$  in  $\Omega^*$ , if there does not exist any pattern  $P$  in  $\Omega^*$  such that  $M_{\text{patt}}(P)$  is greater than  $M_{\text{patt}}(P^*)$  and  $P$  substitutes  $P^*$ .

$$P^* \in \Omega^*, \text{ if } \nexists P \in \Omega^* \mid M_{\text{patt}}(P) > M_{\text{patt}}(P^*) \text{ and } \text{subst}(P, P^*, \tau) = \text{true} \quad (4.6)$$

**Definition 18** (*Joint support*) Given two patterns  $P$  and  $P'$ , if  $P$  substitutes  $P'$  then  $P$  should represent  $P'$  in the graphs where  $P'$  occurs. In  $\Omega^*$ , the occurrence list of  $P$  will contain both the occurrences of  $P$  as well as those of  $P'$  (the occurrence lists are joined). Formally:

$$\forall P, P' \in \Omega, \text{ if } \text{subst}(P, P', \tau) = \text{true} \text{ then } D_P = D_P \cup D_{P'} \quad (4.7)$$

where  $D_P$  and  $D_{P'}$  are correspondingly the occurrence list of  $P$  and that of  $P'$ .

### 4.3.3 Algorithm

Given a set of patterns  $\Omega$  and a substitution matrix  $\mathcal{M}$ , we propose UNSUB-PATT (see Algorithm 1), a pattern selection algorithm which enables detecting the set of unsubstituted patterns  $\Omega^*$  within  $\Omega$ . Based on our similarity concept, all the patterns in  $\Omega^*$  are dissimilar, since it does not contain any pair of patterns that are substitutable. The general process of the algorithm is

---

**Algorithm 1:** UNSUBPATT

---

**Data:**  $\Omega, \mathcal{M}, \tau$   
**Result:**  $\Omega^*$ : {unsubstituted patterns}

```

1 begin
2   divide  $\Omega$  into  $k$  subsets  $\mid \forall P', P'' \in \Omega^k, \mid V_{P'} \mid = \mid V_{P''} \mid$  and
    $\mid E_{P'} \mid = \mid E_{P''} \mid$ ;
3   foreach  $\Omega^k \subset \Omega$  do
4      $\Omega^k \leftarrow \text{sort}(\Omega^k \text{ by } M_{patt})$ ;
5     foreach  $P \in \Omega^k$  do
6       if  $M_{patt}(P) > 0$  then
7         foreach  $P' \in \Omega^k \setminus \{P\} \mid M_{patt}(P') < M_{patt}(P)$  do
8           if  $M_{patt}(P') > 0$  then
9             if  $\text{shape}(P, P')$  then
10              foreach mapping between  $P$  and  $P'$  do
11                if  $\text{subst}(P, P', \tau)$  then
12                   $\text{support}(P) \leftarrow \text{join supports}(P, P')$ ;
13                  remove  $P'$  from  $\Omega^k$ ;
14                  goto 7;
15    $\Omega^* \leftarrow \Omega^* \cup \Omega^k$ ;

```

---

described as follows: first,  $\Omega$  is divided into subsets of patterns having the same number of nodes and edges. In order to preserve the most mutable patterns, each subset is sorted in a descending order by the pattern mutation score  $M_{patt}$ . Then, each subset is browsed starting from the pattern having the highest  $M_{patt}$ . For each pattern, we look for all the other patterns it is able to substitute, with respect to the substitution threshold. The test of substitution is performed iteratively for every possible mapping between pairs of patterns until a substitution is found. If a substitution is found with a particular mapping then there is no need to proceed testing the substitution with the rest of the mappings since we are not looking for the best substitution

but we are only looking for a possible one, with respect to the given threshold. For each pattern, we remove all the patterns it substitutes and we add their supports' lists to that of the preserved pattern such that the latter will represent all the patterns it substitutes wherever they occur. The remaining patterns represent the representative unsubstituted pattern set.

**Property 2** *Let  $\Omega$  be a set of patterns and  $\Omega^*$  its subset of unsubstituted patterns based on a substitution matrix  $\mathcal{M}$  and a threshold  $\tau$ , i.e.,  $\text{UNSUBPATT}(\Omega, \mathcal{M}, \tau) = \Omega^*$ .  $\Omega^*$  can not be summarized by one of its proper subsets but only by itself, with respect to  $\tau$ . Formally:*

$$\text{UNSUBPATT}(\Omega^*, \mathcal{M}, \tau) = \Omega^* \quad (4.8)$$

**Proof 1** *Lets suppose that :*

- **hypothesis 1:**  $\Omega^* \setminus \text{UNSUBPATT}(\Omega^*, \mathcal{M}, \tau) \neq \emptyset$
- **hypothesis 2:**  $\text{UNSUBPATT}(\Omega^*, \mathcal{M}, \tau) \setminus \Omega^* \neq \emptyset$

*Hypothesis 1 supposes that  $\Omega^*$  still contains substitutable patterns. This is impossible, since according to Definition 17, there does not exist any pair of patterns in  $\Omega^*$  that are substitutable. Given a threshold  $\tau$ ,  $\Omega^*$  cannot be summarized by one of its proper subsets but only by itself. Formally:  $\forall P \in \Omega^*, \nexists P' \in \Omega^* | M_{\text{patt}}(P) > M_{\text{patt}}(P')$  and  $\text{subst}(P, P', \tau)$*

*As for hypothesis 2 to be true,  $\text{UNSUBPATT}$  is supposed to generate new patterns that were not originally in  $\Omega^*$ . This contradicts  $\text{UNSUBPATT}$  basics especially Definition 17 since  $\text{UNSUBPATT}$  is supposed to remove substituted patterns, not to generate new ones.*

The minimum description length (MDL) principle [Rissanen 1978, Grünwald 2007] suggests that given a set of observed data, the best explanation is the one that permits the greatest compression of the data. According to the MDL,  $\Omega^*$  represents a reliable summarization of  $\Omega$ .

**Complexity** Suppose  $\Omega$  contains  $n$  patterns.  $\Omega$  is divided into  $g$  groups, each containing patterns of order  $k$ . This is done in  $O(n)$ . Each group  $\Omega^k$  is sorted in  $O(|\Omega^k| * \log|\Omega^k|)$ . Searching for unsubstituted patterns requires browsing  $\Omega^k$  ( $O(|\Omega^k|)$ ) and for each pattern, browsing in the worst case all remaining patterns ( $O(|\Omega^k|)$ ) to check the shape  $O(k)$  and the substitution  $O(k)$ . This means that searching for unsubstituted patterns in a group  $\Omega^k$  can be done in  $O(|\Omega^k|^2 * k^2)$ . Hence, in the worst case, the complexity of our algorithm is  $O(g * m_{\text{max}}^2 * k_{\text{max}}^2)$ , where  $k_{\text{max}}$  is the maximum pattern order and  $m_{\text{max}}$  is the number of patterns of the largest group  $\Omega^k$ .

#### 4.3.4 Illustrative example

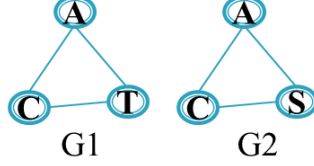


Figure 4.3: An example of two substitutable subgraphs. <sup>1</sup>

Given the following toy subgraphs G1 and G2 (see Figure 4.3), we want to check if G1 and G2 are substitutable, and if so, which subgraph will be considered as the representative one. The substitution test is based on the definitions in section 4.3.2 using the substitution matrix Blosum62. The general substitution test is processed as follows:

- For each substitution score  $x \in Blosum62$ ,  $x \leftarrow e^x$
- Elementary conservation score (we compute the elementary conservation scores for all nodes of G1 and G2):
  - $M_{el}(A) = \frac{\mathcal{M}(A,A)}{\sum_{i=1}^{20} \mathcal{M}(A,l_i)} \simeq 0.840$
  - $M_{el}(C) = \frac{\mathcal{M}(C,C)}{\sum_{i=1}^{20} \mathcal{M}(C,l_i)} \simeq 0.999$
  - $M_{el}(T) = \frac{\mathcal{M}(T,T)}{\sum_{i=1}^{20} \mathcal{M}(T,l_i)} \simeq 0.936$
  - $M_{el}(S) = \frac{\mathcal{M}(S,S)}{\sum_{i=1}^{20} \mathcal{M}(S,l_i)} \simeq 0.776$
- Pattern mutation score:
  - $M_{patt}(G1) = 1 - (M_{el}(A) * M_{el}(C) * M_{el}(T)) \simeq 1 - (0.840 * 0.999 * 0.936) \simeq 1 - 0.786 \simeq 0.214$
  - $M_{patt}(G2) = 1 - (M_{el}(A) * M_{el}(C) * M_{el}(S)) \simeq 1 - (0.840 * 0.999 * 0.776) \simeq 1 - 0.652 \simeq 0.348$
  - Thus  $M_{patt}(G1) < M_{patt}(G2)$
- Structural isomorphism:  $shape(G1, G2) = true$ . This function check if G1 and G2 are isomorphic and returns all possible mappings between them.

---

<sup>1</sup>The subgraphs are considered as substitutable for all substitution thresholds  $\tau \lesssim 68.32\%$  using Blosum62 as substitution matrix.

In UNSUBPATT, we compute the substitution score for every possible mapping between  $G1$  and  $G2$ , until a substitution score with a value greater or equal to the given substitution threshold is found or no other mapping is possible. Here, we only show, as an example, how the substitution score is computed for only one mapping between  $G1$  and  $G2$  among the possible ones. The considered mapping for this example is:  $A \leftrightarrow A$ ,  $C \leftrightarrow C$ ,  $S \leftrightarrow T$ .

- Pattern substitution score:

$$- S_{patt}(G2, G1) = \frac{(\frac{\mathcal{M}(A,A)}{\mathcal{M}(A,A)} + \frac{\mathcal{M}(C,C)}{\mathcal{M}(C,C)} + \frac{\mathcal{M}(S,T)}{\mathcal{M}(S,S)})}{|G2|} \simeq 0.6832$$

- Thus,  $G2$  substitutes  $G1$  for all substitution thresholds  $0\% \leq \tau \leq 68.32\%$

If the user-specified substitution threshold is greater than 68.32% (*i.e.*,  $\forall \tau \geq 68.32\%$ ) then UNSUBPATT proceed checking the other possible mappings. Otherwise (*i.e.*,  $\forall \tau \leq 68.32\%$ ),  $G2$  substitutes  $G1$ . In this case, supports of  $G2$  and  $G1$  are joined then  $G1$  is removed:

- Joining support:  $D_{G2} = D_{G2} \cup D_{G1}$  ( $D_{Gi}$  is the occurrence list of  $Gi$ )
- Remove  $G1$

## 4.4 Experiments

### 4.4.1 Datasets

In order to experimentally evaluate our approach, we use four datasets of protein 3D-structures, which also have been used in [Yan 2008] and [Fei 2010]. Each dataset consists of two classes equally divided into positive and negative samples. Positive samples are proteins selected from a considered protein family whereas negative samples are proteins randomly gathered from the Protein Data Bank [Berman 2000]. Table 4.1 summarizes the characteristics of each dataset:

- SCOP ID: identifier of the protein family in SCOP [Andreeva 2008]
- $|G|$ : number of proteins in the dataset
- Avg. $|V|$ : average number of nodes
- Avg. $|E|$ : average number of edges
- Max. $|V|$ : maximal number of nodes



-  $\text{Max.}|E|$ : maximal number of edges

**G-proteins** : DS1 contains protein 3D-structures from the G-protein family, also known as guanine nucleotide-binding proteins. These proteins are mainly involved in transmitting chemical signals originating from outside a cell into the inside of it. G-proteins are able to activate a cascade of further signaling events resulting a change in cell functions. They regulate metabolic enzymes, ion channels, transporter, and other parts of the cell machinery, controlling transcription, motility, contractility, and secretion, which in turn regulate diverse systemic functions such as embryonic development, learning and memory, and homeostasis.

**C1-set domains** : The C1-set domains composing DS2 are immunoglobulin-like domains, similar in structure and sequence. They resemble the antibody constant domains. They are mostly found in molecules involved in the immune system, in the major histocompatibility complex class I and II complex molecules, and in various T-cell receptors.

**C-type lectin domains** : Lectins occur in plants, animals, bacteria and viruses. In DS3, the C-type (Calcium-dependent) lectins are family of lectins which share structural homology in their high-affinity carbohydrate-recognition domains. There are at least twelve structural families of lectins, of which C-type lectins is one. This family involves groups of proteins playing divers functions including cell-cell adhesion, immune response to pathogens and apoptosis.

**Protein kinases, catalytic subunit** : Protein kinases, catalytic subunit composing DS4 play a role in various cellular processes, including division, proliferation, apoptosis, and differentiation. They are mainly proteins that modifies other ones by chemically adding phosphate groups to them. This usually results in a functional change of the target protein by changing enzyme activity, cellular location, or association with other proteins. The catalytic subunits of protein kinases are highly conserved, and several structures have been solved, leading to large screens to develop kinase-specific inhibitors for the treatments of a number of diseases.

Table 4.1: Characteristics of the experimental datasets

Dataset	SCOP ID	Family name	$ G $	$\text{Avg.} V $	$\text{Avg.} E $	$\text{Max.} V $	$\text{Max.} E $
DS1	52592	G-proteins	66	246	971	897	3 544
DS2	48942	C1-set domains	76	238	928	768	2 962
DS3	56437	C-type lectin domains	76	185	719	755	3 016
DS4	88854	Protein kinases, catalytic subunit	82	275	1077	775	3 016

Proteins are parsed into graphs of amino acids using the main atom method (see section 2.3.4.2). Each node represents an amino acid residue and is labeled with its amino acid type. Two nodes  $u$  and  $v$  are linked by an edge  $e(u, v) = 1$  if the euclidean distance between their two  $C_\alpha$  atoms  $\Delta(C_\alpha(u), C_\alpha(v))$  is below a threshold distance  $\delta$ . In the literature, many methods use this definition with usually  $\delta \geq 7\text{\AA}$  on the argument that  $C_\alpha$  atoms define the overall shape of the protein conformation [Huan 2005]. In our experiments, we use  $\delta = 7\text{\AA}$ .

#### 4.4.2 Protocol and settings

Generally, in a pattern selection approach two aspects are emphasized, namely the number of selected patterns and their interestingness. In order to evaluate our approach, we first use the state-of-the-art method of frequent subgraph discovery gSpan [Yan 2002] to find the frequent subgraphs in each dataset with a minimum frequency threshold of 30%. Then, we use UNSUBPATT to select the unsubstituted patterns among them with a minimum substitution threshold  $\tau=30\%$  and *Blosum62* (see Table 2.2) as the substitution matrix. We use *Blosum62* because it turned out that it performs well on detecting the majority of weak protein similarities [Eddy 2004], and it is used as the default matrix by most biological applications such as BLAST [Altschul 1990]. It is worth mentioning that the choice of 30% as minimum frequency threshold for frequent subgraph extraction is to have fewer patterns in order to make the experimental evaluation feasible due to time and computational limitations.

In order to evaluate the number of selected subgraphs, we define the selection rate as the rate of the number of unsubstituted subgraphs from the initial set of frequent subgraphs. Formally :

$$\text{Selection rate} = \frac{|\Omega^*| * 100}{|\Omega|} \quad (4.9)$$

To evaluate the interestingness of the selected patterns, we use them as features for classification. We perform a 5-fold cross-validation classification (5 runs) on each protein dataset. We encode each protein into a binary vector, denoting by "1" or "0" the presence or the absence of the feature in the considered protein. For classification, we use classifiers from the workbench Weka [Witten 2005].

## 4.5 Results and discussion

In this section, we conduct experiments to examine the effectiveness and efficiency of UNSUBPATT in finding the representative unsubstituted subgraphs.

Table 4.2: Number of frequent subgraphs ( $\Omega$ ), representative unsubstituted subgraphs ( $\Omega^*$ ) and the selection rate

Dataset	$ \Omega $	$ \Omega^* $	Selection rate (%)
DS1	799094	7291	0.91
DS2	258371	15898	6.15
DS3	114792	14713	12.82
DS4	1073393	9958	0.93

Moreover, we test the effect of changing the substitution matrix and the substitution threshold on the results. We further study the size-based distribution of patterns and we compare the classification results of our approach with those of other subgraph selection methods from the literature.

#### 4.5.1 Empirical results

We show the results of our experiments in terms of number of patterns and classification results. The obtained average results are reported in the Tables 4.2 and 4.3.

The high number of discovered frequent subgraphs is due to their combinatorial nature. It may increase or decrease depending on the number of graphs, their density and mainly on the similarity between graphs since the more similar they are, the more common fragments they would have. The results reported in Table 4.2 show that our approach decreases considerably the number of subgraphs. The selection rate shows that the number of unsubstituted patterns  $|\Omega^*|$  does not exceed 13% of the initial set of frequent subgraphs  $|\Omega|$  in the worst case with DS3 and even reaches less than 1% with DS1 and DS4. This proves that exploiting the domain knowledge by incorporating, in our case, the substitution matrix in the selection enables detecting many similarities between patterns that are possibly ignored by current subgraph selection approaches.

The classification results using naive bayes (NB) is reported in Table 4.3. They help evaluating the quality of the selected patterns. Indeed, they will demonstrate if the unsubstituted patterns are really representative or arbitrarily selected. Table 4.3 shows that the classification accuracy significantly increases with all datasets. We notice a huge leap in accuracy especially with DS1 and DS4 with a gain of more than 17% and reaching almost full accuracy with DS4. To better understand the accuracy results, we report the average precision, recall, F-measure and AUC values for all cases. We also notice an enhancement of performance with all the mentioned quality metrics. This

Table 4.3: Accuracy, precision, recall (sensitivity), F-score and AUC of the classification of each dataset using NB coupled with frequent subgraphs (FSg) then representative unsubstituted subgraphs (USP)

Dataset	Accuracy		Precision		Recall		F-score		AUC	
	FSg	USP	FSg	USP	FSg	USP	FSg	USP	FSg	USP
DS1	0.62	0.78	0.61	0.69	0.70	0.90	0.64	0.78	0.64	0.78
DS2	0.80	0.90	0.86	0.94	0.74	0.86	0.79	0.89	0.79	0.89
DS3	0.86	0.94	0.89	1.00	0.86	0.89	0.86	0.94	0.86	0.94
DS4	0.79	0.98	0.86	0.92	0.70	0.98	0.76	0.94	0.76	0.94

supports the reliability of our selection.

#### 4.5.2 Results using other substitution matrices

Besides Blosum62, biologists also defined other substitution matrices describing the likelihood that two amino acid types would mutate to each other in evolutionary time. We want to study the effect of using other substitution matrices on the experimental results. Hence, we perform the same experiments following the same protocol and settings but using two other substitution matrices, namely *Blosum80* and *Pam250*. The results are reported in Table 4.4. We compare the obtained results in terms of number of subgraphs and classification accuracy with those obtained using the whole set of frequent subgraphs and those using subgraphs previously selected by UNSUBPATT with Blosum62. A high selection rate accompanied with a clear enhancement of the classification accuracy is noticed using UNSUBPATT with all the substitution matrices. It is clearly noticed that even using different substitution matrices, UNSUBPATT is able to select a small yet relevant subset of patterns. It is also worth mentioning that for all the datasets, the best classification accuracy is obtained using Blosum62 and the best selection rate is achieved using Pam250. This is simply due to how distant proteins within the same dataset are, since each substitution matrix was constructed to implicitly express a particular theory of evolution.

#### 4.5.3 Impact of varying the substitution threshold

In our experiments, we used a substitution threshold (of 30%) to select the unsubstituted patterns from the set of discovered frequent subgraphs. Here, we study the impact of varying the substitution threshold on both the number of selected subgraphs and the classification results. We perform the same

Table 4.4: Number of subgraphs (#SG) and accuracy (Acc) of the classification of each dataset using NB coupled with frequent subgraphs (FSg) then representative unsubstituted subgraphs using Blossum62 (USP<sub>62</sub>), Blossum80 (USP<sub>80</sub>) and Pam250 (USP<sub>250</sub>)

Dataset	FSg		USP <sub>62</sub>		USP <sub>80</sub>		USP <sub>250</sub>	
	#SG	Acc	#SG	Acc	#SG	Acc	#SG	Acc
DS1	799094	0.62	7291	0.78	7328	0.67	6137	0.68
DS2	258371	0.80	15898	0.90	15930	0.87	15293	0.87
DS3	114793	0.86	14713	0.94	14792	0.91	14363	0.93
DS4	1073393	0.79	9958	0.98	10417	0.90	9148	0.90

experiments following the same protocol and settings while varying the substitution threshold from 0% to 90% with a step-size of 10. Figure 4.4 presents the selection rate for all substitution thresholds. In order to check if the enhancements of results are due to our selected patterns or to the classifier, we perform the same experiments using naive bayes (NB) (Figure 4.5) and two other well-known classifiers namely the support vector machine (SVM) (Figure 4.6) and decision tree (C4.5) (Figure 4.7). The classification accuracy of the initial set of frequent subgraphs (gSpan) is considered as a standard value for comparison. Thus, the accuracy values of UNSUBPATT that are above the line of the standard value are considered as gains, and those under the standard value are considered as losses.

In Figure 4.4, we notice that UNSUBPATT reduces considerably the number of patterns especially with lower substitution thresholds. In fact, the number of representative unsubstituted patterns does not exceed 50% for all substitution thresholds below 80% and even reaches less than 1% in some cases. This important reduction in the number of patterns comes with a notable enhancement of the classification accuracies over all datasets.

Figures 4.5, 4.6 and 4.7 show that the unsubstituted patterns allow better classification performance compared to the original set of frequent subgraphs. UNSUBPATT scores very well with the three used classifiers and even reaches full accuracy in some cases. Overall, the same behavior is noticed with the three datasets. A cross view over the figures is possible, showing that there is no bias of the datasets nor of the classifier. This confirms our assumptions and shows that our selection is reliable and contributes to the enhancement of the accuracy.

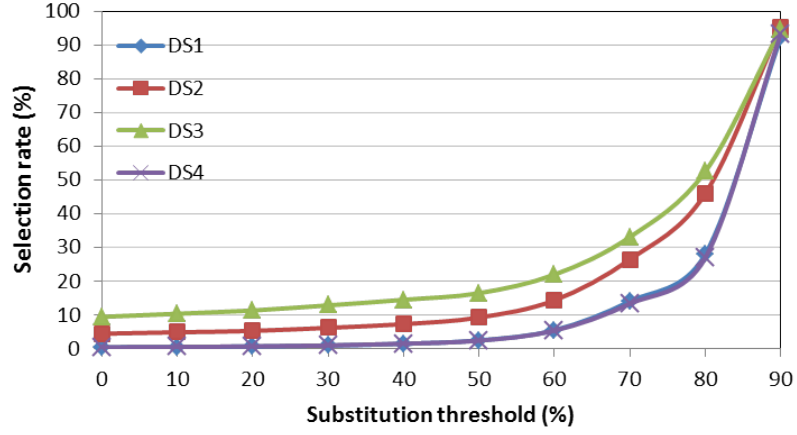


Figure 4.4: Rate of representative unsubstituted patterns  $\Omega^*$  from all frequent subgraphs  $\Omega$  depending on the substitution threshold ( $\tau$ ).

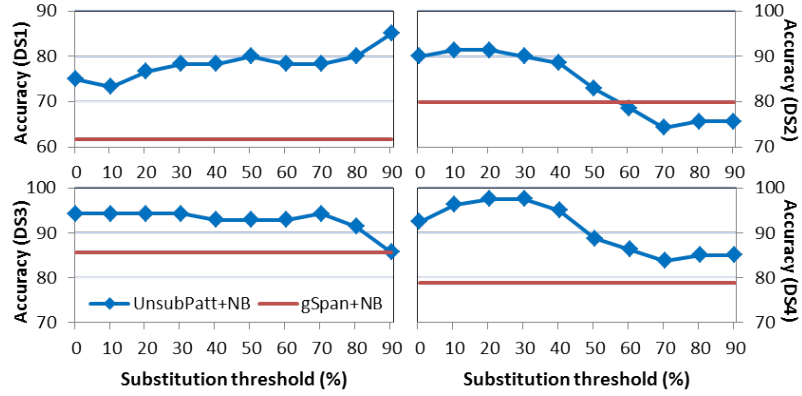


Figure 4.5: Classification accuracy of the four protein datasets using naive bayes (NB) and all frequent subgraphs (gSpan+NB) then the representative unsubstituted ones with different substitution thresholds (UNSUBPATT+NB).

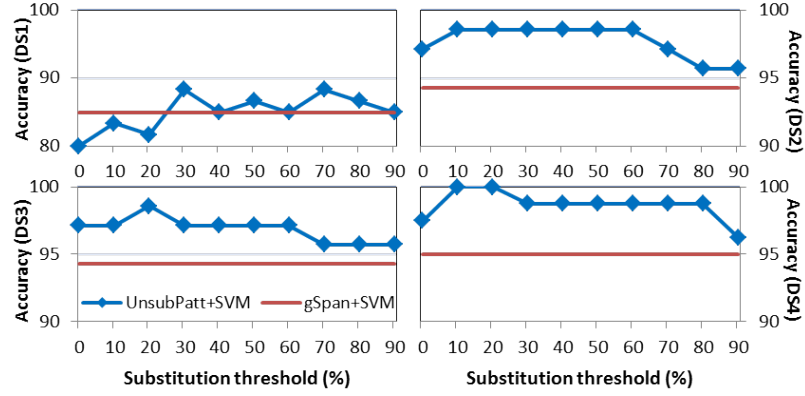


Figure 4.6: Classification accuracy by support vector machine (SVM).

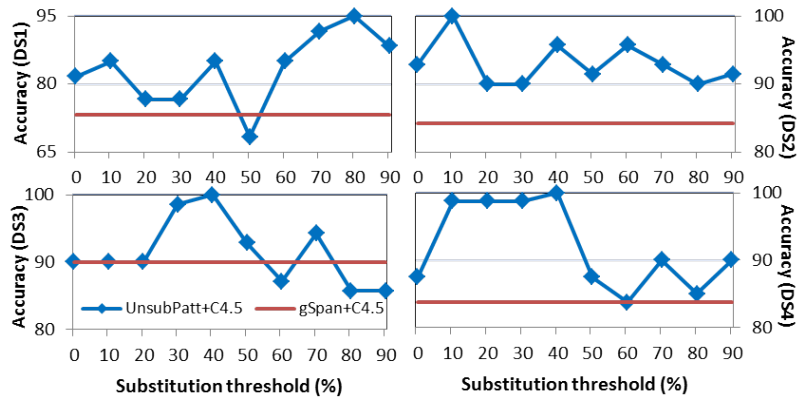


Figure 4.7: Classification accuracy by decision trees (C4.5).

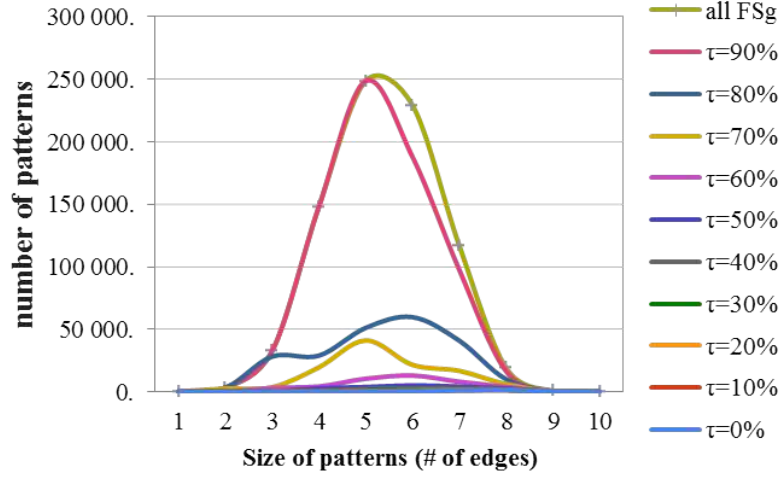


Figure 4.8: Distribution of patterns of DS1 for all the frequent subgraphs and for the representative unsubstituted ones with different substitution thresholds (left: standard arithmetic scale. Right: logarithmic scale.).

#### 4.5.4 Smoothing the distribution of patterns

In this section, we study the distribution of patterns based on their size (number of edges). We try to check which sizes of patterns are more concerned by the selection. Figures 4.8, 4.9, 4.10 and 4.11 draw the distribution of patterns for the original set of frequent subgraphs and for the final set of representative unsubstituted ones with all the substitution thresholds using Blosum62. The downward tendency of UNSUBPATT using lower substitution thresholds and with respect to the original set of frequent subgraphs is very clear. In fact, an effect of smoothing is clearly noticed over the whole sets, since UNSUBPATT leans towards cutting off the peaks and flattening the curves with lower substitution thresholds. The same tendency is also noticed with all the datasets. Another interesting observation is that the curves are flattened in the regions of small subgraphs as well as in the regions of large and dense ones. This demonstrates the effectiveness of UNSUBPATT with both small and large subgraphs.

#### 4.5.5 Comparison with other approaches

Here, we compare UNSUBPATT with some of the current subgraph selection approaches. Figure 4.12 illustrates the classification accuracies of UNSUBPATT besides those of the other approaches from the literature namely LEAP [Yan 2008], gPLS [Saigo 2008], COM [Jin 2009] and LPGBCMP [Fei 2010].



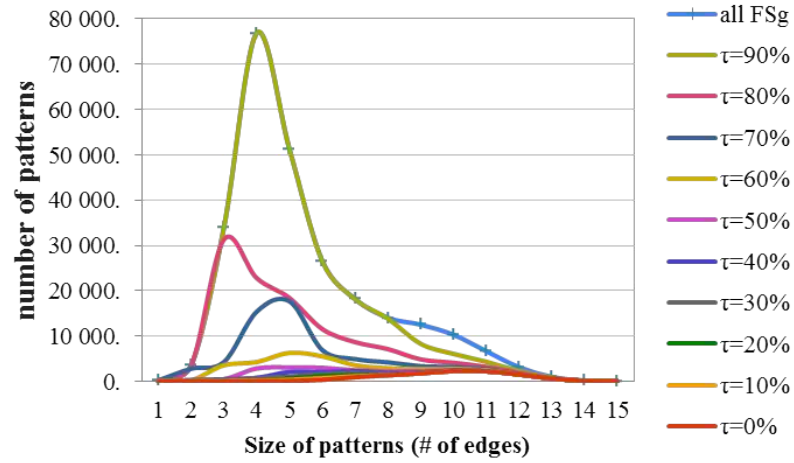


Figure 4.9: Distribution of patterns of DS2 for all the frequent subgraphs and for the representative unsubstituted ones with different substitution thresholds (left: standard arithmetic scale. Right: logarithmic scale.).

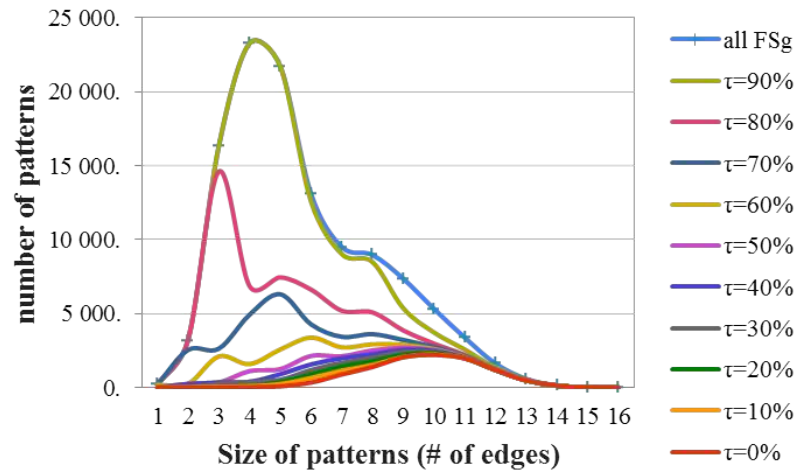


Figure 4.10: Distribution of patterns of DS3 for all the frequent subgraphs and for the representative unsubstituted ones with different substitution thresholds (left: standard arithmetic scale. Right: logarithmic scale.).

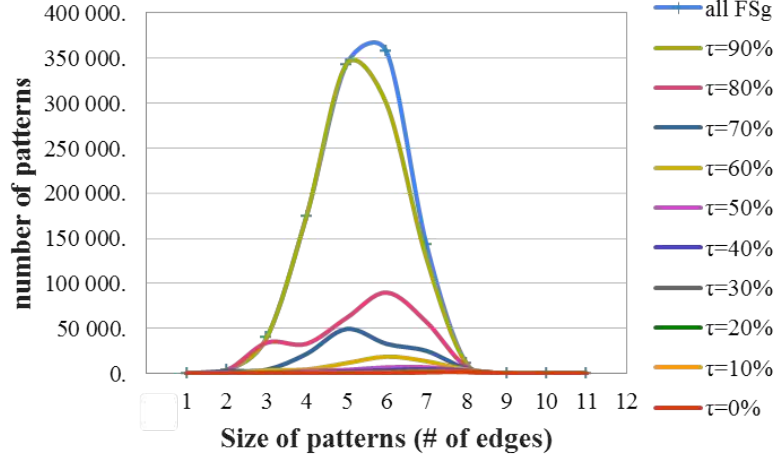


Figure 4.11: Distribution of patterns of DS4 for all the frequent subgraphs and for the representative unsubstituted ones with different substitution thresholds (left: standard arithmetic scale. Right: logarithmic scale.).

As these approaches were originally tested on the same datasets used in our experiments, we report the results of each approach with the parameters recommended from their authors. For UNSUBPATT, we report the results using a substitution threshold  $\tau = 30\%$ , Blosum62 and SVM. We also report UNSUBPATT<sub>max</sub> as the best accuracies among all the substitution thresholds (used in Figure 4.6). For LEAP+SVM, LEAP is used iteratively to discover discriminative subgraphs with a leap length=0.1. The discovered subgraphs are considered as features to train SVM with a 5-fold cross validation. COM is used with  $t_p = 30\%$  and  $t_n = 0\%$ . For gPLS, the frequency threshold is 30% and the best accuracies are reported among all parameters combinations for  $m = 2, 4, 8, 16$  and  $k = 2, 4, 8, 16$ , where  $m$  is the number of iterations and  $k$  is the number of patterns per search. For LPGBCMP, threshold values of  $max_{var} = 1$  and  $\delta = 0.25$  were respectively used for feature consistency map building and for overlapping. As we are testing on protein classification, it would be also interesting to compare with tool of biologists. We performed an alignment-based classification over the four datasets. One is sequential (using protein primary structure in FASTA format (see Section A.2)) using Blast [Altschul 1990] and the other is structural (using protein 3D-structure) using Dali [Holm 2010]. For each dataset, we make an alignment of each protein against all the others. We assign to the query protein the class of the reference protein with the best hit score.

Although Dali-based classification represents the most competitive ap-

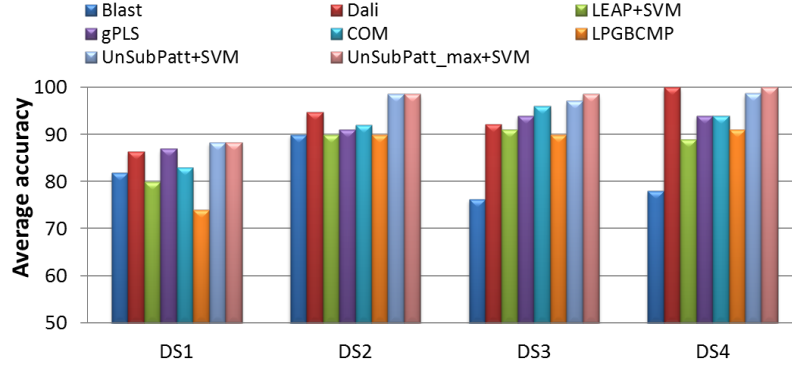


Figure 4.12: Classification accuracy comparison with Blast, Dali and pattern selection approaches.

proach over Blast and the other pattern-based classifications, UNSUBPATT outperforms them all in the four cases except for DS4 when UNSUBPATT<sub>max</sub> and Dali reached both full accuracy. This proves that UNSUBPATT represents a very competitive and promising approach and that using the substitution between amino acids allows it to select a significant and very informative subset of patterns.

#### 4.5.6 Runtime analysis

To study the variation of UNSUBPATT's runtime with larger amounts of data, we use different sets of frequent patterns from 10000 to 100000 with step-size of 10000. In Table 4.5, we report the runtime results for the pattern sets using three substitution thresholds: 10%, 30% and 50%.

Even though the complexity of the problem due to the combinatorial test of substitution between subgraphs, our algorithm scales well with higher amounts of data. Although the number of patterns increases, the runtime stills reasonable and not exponential in real world cases. This can be explained by the fact that the complexity of UNSUBPATT is equal to  $O(g * m_{max}^2 * k_{max}^2)$  in the worst case (as previously explained in 15), where  $g$  is the number of groups,  $k_{max}$  is the order of the largest pattern and  $m_{max}$  is the number of patterns of the largest group  $\Omega^k$ . In practice, many patterns are substituted and thus removed during the selection. Hence, in most real world cases, the complexity of UNSUBPATT is equal to  $O(g * (m_{max} * \log(m_{max})) * (k_{max} * \log(k_{max})))$ .

The use of different substitution thresholds slightly affected the runtime of UNSUBPATT, since the numbers of selected patterns are comparable for all thresholds.

A possible way to make UNSUBPATT runs faster is parallelization. In fact,

Table 4.5: Runtime analysis of UNSUBPATT with different substitution thresholds

Number of patterns	Substitution thresholds		
	$\tau = 10\%$	$\tau = 30\%$	$\tau = 50\%$
10000	4s	4s	4s
20000	8s	8s	10s
30000	13s	13s	17s
40000	18s	18s	25s
50000	23s	23s	33s
60000	28s	28s	41s
70000	35s	35s	52s
80000	40s	42s	66s
90000	46s	49s	80s
100000	53s	57s	136s

UNSUBPATT can be easily parallelized, since it tests separately the substitution among each group of subgraphs having the same size and order. Hence, these groups can be distributed and treated separately in different processes.

## 4.6 Conclusion

In this chapter, we proposed a novel selection approach for mining a representative subset of patterns from a set of frequent subgraphs. Unlike current methods that are based on the relations between patterns in the transaction space, our approach considers the distance between patterns in the pattern space. Experimental results revealed the importance of incorporating the prior domain knowledge and showed that using the information of substitution between amino acids allowed UNSUBPATT to detect many similarities between patterns that current subgraph selection approaches ignore. UNSUBPATT is able to considerably reduce the number of subgraphs by selecting a more representative and informative subset enabling easier and more efficient further explorations. UNSUBPATT can also be used on protein sequences (seen as line graphs) and it is unsupervised which allows it to be used in different mining tasks and in other motif-based analysis.

It is also worth mentioning that UNSUBPATT is not limited to protein 3D-structures but can be generalized to other types of data whenever it is possible to define a matrix representing the similarity between the nodes labels.

A promising future direction is to consider also the insertions and deletions over patterns with different sizes. Although this increases exponentially

the complexity and the difficulty of the selection, it is closer to the real world substitution phenomenon. Another interesting future work could be to embed the selection within the extraction process in order to directly mine the representative patterns from data. This is further discussed in Section 6.4.1.2.

This chapter was the subject of a number of publications, namely a poster paper in ACM BCB [Dhifi 2012b], a conference paper at JOBIM [Dhifi 2013c] and a journal paper in JCB [Dhifi 2013b]. It was also the subject of two oral presentations given at MLCB [Dhifi 2012a] and JFD [Dhifi 2013a].

# TRS : Towards an efficient discovery of topological representative subgraphs

---

## Contents

---

<b>5.1</b>	<b>Aims</b>	<b>87</b>
<b>5.2</b>	<b>Introduction</b>	<b>88</b>
<b>5.3</b>	<b>Top-k topological representative subgraph selection</b>	<b>90</b>
5.3.1	Problem Statement	90
5.3.2	Naïve approach	90
5.3.3	Topological representative subgraph selection	91
<b>5.4</b>	<b>Experimental analysis</b>	<b>97</b>
5.4.1	Datasets	97
5.4.2	Protocol and settings	98
<b>5.5</b>	<b>Results and discussion</b>	<b>99</b>
5.5.1	Empirical results	99
5.5.2	Size-based distribution of patterns	101
5.5.3	Runtime analysis	103
<b>5.6</b>	<b>Conclusion</b>	<b>107</b>

---

## 5.1 Aims

In the previous chapter, we proposed a novel approach, called UNSUBPATT, for subgraph selection that incorporates the prior domain knowledge that are often ignored by current subgraph selection methods. Precisely, it uses the substitution information between the nodes labels (amino acid types) to detect similarities between subgraphs. In other words, the similarity between subgraphs is purely semantic as it depends on the similarities between nodes

labels that is defined in the substitution matrix. In this chapter, we introduce another subgraph selection approach that focuses on the structural similarity rather than the semantic similarity. Unlike existing structural-based selection approaches that look into every single detail, this approach considers the overall topological similarity between subgraphs by means of a set of topological descriptors. This makes it easily extendable with a user-specified set of descriptors depending on the application and the sought information.

## 5.2 Introduction

Feature selection for graph data is a way to tackle the dimensionality problem when the number of frequent subgraphs is very high. As structural similarity represents one major cause of redundancy in frequent subgraphs, many works have been proposed for subgraph selection based on exact or approximate structural similarity [Yan 2003, Thomas 2006, Hasan 2007, Chen 2008]. Two pioneer works that fall in this type are [Yan 2003] for mining closed subgraphs and [Thomas 2006] for mining maximal subgraphs. In both works, only the closed or maximal subgraphs are maintained and the rest of frequent subgraphs are removed. Many works have been proposed based on closed and maximal subgraphs such as [Takigawa 2011, Li 2007]. Although the set of closed or maximal subgraphs is much smaller than the set of frequent ones, the number of subgraphs is still very high in real-world cases.

Many works have been proposed for subgraph selection based on approximate structural similarity. In [Hasan 2007], authors proposed an approach for subgraphs extraction and selection. For selection, the structural similarity between two subgraphs is measured by how much does their maximum common subgraph [Abu-Khzam 2007] represents from their overall structure. A very close work is [Chen 2008], where authors proposed an approach for mining a set of structural representative subgraphs among the frequent ones. They adopted a two-step approach that is based on approximate structural similarity on micro and macro sides. In the first step, they consider a tolerance threshold to summarize approximately isomorphic subgraphs into one representative. In the second step, they collapse multiple structurally similar subgraphs into one representative using a clustering algorithm.

Existing selections approaches that are based on exact or approximate structural similarity, look into every single detail and test the structural similarity of subgraphs by establishing a matching between them. This similarity detection strategy is not efficient in many real-world applications. On one hand, because the combinatorial nature of graphs makes computing every possible matching between pairs of subgraphs very costly. On the other hand,

exact and even approximate structural similarity are not efficient enough to detect all similar subgraphs in real-world data. Indeed, exact structural similarity does not allow detecting similar yet slightly different subgraphs, and approximate structural similarity has the problem of threshold setting. Since a tight threshold will prevent detecting many similar subgraphs that slightly differ in structure beyond the tolerance threshold and thus preserve a high number of subgraphs. In contrast, a loose threshold will hinder the soundness of the selection because of false positives. This rises the need for a different way to consider the structural similarity such that both close and distant structural similarities would be detected with respect to the soundness of results.

Considering topological properties instead of exact or approximate structural isomorphism was inspired by works like [Rodenacker 1990, Leskovec 2005, Veeramalai 2008, Li 2012, Ranu 2012, Tong 2012, Gibert 2012] where authors showed the importance and efficiency of topological attributes in describing graph data. For instance, in [Li 2012], authors proposed a classification framework based on the assumption that graphs belonging to the same class have similar topological descriptions. Our approach is based on similar assumption and consider that structurally similar subgraphs should have similar topological properties such that even a slight difference does not affect the overall topological similarity. Besides, depending on the application context, a user may be interested only in some specific structural properties. However, considering exact or approximate structural similarity approaches does not allow this specificity.

In order to overcome these drawbacks and to select a small yet structurally non-redundant set of subgraphs, we propose a novel approach that mines the top-k topological representative subgraphs among the frequent ones. At a glance, our approach involves two steps. In the first step, each subgraph is encoded into a topological description-vector containing the corresponding values for a set of topological attributes. In the second step, subgraphs with similar topological descriptions are clustered together and the central subgraph in each cluster is considered as the representative delegate. Our approach overcomes the costly isomorphism needed to perform the exact or approximate structural similarity and allows detecting hidden similarities like spectral radius or closeness centrality, that exact or approximate structural similarity approaches are unable to detect. Besides, our approach can be easily extended by enabling the user to target a specific set of topological attributes depending on how important each one is to the application.



## 5.3 Top-k topological representative subgraph selection

### 5.3.1 Problem Statement

Even though the existing approaches for subgraph selection greatly enhanced the selection process, the number of selected subgraphs is still high. Yet, we want to show as few subgraphs as possible so that the user's reviewing efforts are minimized. The general framework of our selection strategy is as follows. Given a set of frequent subgraphs  $\Omega$  and an integer  $k \in [1..|\Omega|]$ , we want to select up to  $k$  representative subgraphs  $\Omega_k \subseteq \Omega$  such that each frequent subgraph  $g \in \Omega$  has one representative subgraph-delegate  $g' \in \Omega_k$ , and each representative subgraph is the closest one to all the subgraphs it represents. To do so, the set of frequent subgraphs is divided into  $k$  clusters using a clustering algorithm, then the cluster centroids are selected to be the representative subgraph-delegates such that each centroid is representative for all subgraphs within the same cluster.

### 5.3.2 Naïve approach

As we are attempting to select top-k representative subgraphs based on clustering, a fundamental part in our selection framework is the graph encoding which consists in the transformation of each subgraph into a different format that is accepted by the clustering algorithm. A naïve solution is to transform the input subgraphs into a context-matrix where each subgraph is represented by a binary vector denoting by 1 or 0 the presence or the absence of the subgraph in each graph in the database. After that, the context-matrix is considered as input for clustering (see Algorithm 2).

---

**Algorithm 2:** NAÏVE APPROACH

---

**Data:** Frequent subgraphs  $\Omega$ , number of representatives  $k$

**Result:** Representative subgraphs  $\Omega^* = \{g_1, g_2, \dots, g_k\}$

1 **begin**

2      $M \leftarrow \cup_{i=1}^{|\Omega|} V_i$ : each subgraph  $g_i \in \Omega$  is encoded into a binary vector  $V_i$  denoting by 1 or 0 correspondingly the presence or the absence of the subgraph in each graph in the database;

3      $\Omega^* \leftarrow \text{Clustering}(M, k)$ ;

4 **end**

---

### 5.3.3 Topological representative subgraph selection

The main idea of our approach is based on the assumption that structurally similar subgraphs should have similar topological properties such that even a slight difference in the structure does not affect the overall similarity [Ingram 2006, Knabe 2008, Li 2012]. Accordingly, we adopt a two-step selection framework, where in the first step we encode each subgraph into a topological description-vector containing the corresponding values for a set of topological attributes. In the second step, we perform a clustering using the topological description-vectors in order to select one representative subgraph delegate from each set of topologically similar subgraphs.

#### 5.3.3.1 Topological attributes

In the first step of our approach each subgraph is encoded into a topological description-vector. We select a set of topological attributes from the literature [Li 2012, Leskovec 2005] that are interesting and efficient in describing connected graphs. In the following, we list and define the selected attributes:

1. **Number of nodes:** The total number of nodes in the graph, also called the graph order  $|V|$ .
2. **Number of edges:** The total number of edges in the graph, also called the graph size  $|E|$ .
3. **Average degree:** The degree of a node  $u$ , denoted  $deg(u)$ , represents the number of nodes adjacent to  $u$ . The average degree of a graph  $G$  is the average value of the degrees of all nodes in  $G$ . Formally:  $deg(G) = \frac{1}{n} \sum_{i=1}^n deg(u_i)$  where  $deg(u_i)$  is the degree of the node  $u_i$  and  $n$  is the number of nodes in  $G$ .
4. **Density:** The density of a graph  $G = (V, E)$  measures how many edges are in  $E$  compared to the maximum possible number of edges between the nodes in  $V$ . Formally:  $den(G) = \frac{2|E|}{(|V|*(|V|-1))}$ .
5. **Average clustering coefficient:** The clustering coefficient of a node  $u$ , denoted by  $c(u)$ , measures how complete the neighborhood of  $u$  is, *i.e.*,  $c(u) = \frac{2e_u}{k_u(k_u-1)}$  where  $k_u$  is the number of neighbors of  $u$  and  $e_u$  is the number of connected pairs of neighbors. If all the neighbor nodes of  $u$  are connected, then the neighborhood of  $u$  is complete and we have a clustering coefficient of 1. If no nodes in the neighborhood of  $u$  are connected, then the clustering coefficient is 0. The average clustering coefficient of an entire graph  $G$  having  $n$  nodes, is given as the average value over all the nodes in  $G$ . Formally:  $C(G) = \frac{1}{n} \sum_{i=1}^n c(u_i)$ .

6. **Average effective eccentricity:** For a node  $u$ , the effective eccentricity represents the maximum length of the shortest paths between  $u$  and every other node  $v$  in  $G$ , i.e.,  $e(u) = \max\{d(u, v) : v \in V\}$ . If  $u$  is isolated then  $e(u) = 0$ . The average effective eccentricity is defined as  $Ae(G) = \frac{1}{n} \sum_{i=1}^n e(u_i)$ , where  $n$  is the number of nodes of  $G$ .
7. **Effective diameter:** The effective diameter represents the maximum value of effective eccentricity over all nodes in the graph  $G$ , i.e.,  $diam(G) = \max\{e(u) \mid u \in V\}$  where  $e(u)$  represents the effective eccentricity of  $u$  as defined above.
8. **Effective radius:** The effective radius represents the minimum value of effective eccentricity over all nodes in the graph  $G$ , i.e.,  $rad(G) = \min\{e(u) \mid u \in V\}$  where  $e(u)$  represents the effective eccentricity of  $u$ .
9. **Closeness centrality:** The closeness centrality measures how fast information spreads from a given node to other reachable nodes in the graph. For a node  $u$ , it represents the reciprocal of the average shortest path length between  $u$  and every other reachable node in the graph, i.e.,  $C_c(u) = \frac{n-1}{\sum_{v \in \{V \setminus u\}} d(u, v)}$  where  $d(u, v)$  is the length of the shortest path between the nodes  $u$  and  $v$ . For a graph  $G$ , we consider the average value of closeness centrality of all the nodes, i.e.,  $C_c(G) = \frac{1}{n} \sum_{i=1}^n C_c(u_i)$ .
10. **Percentage of central nodes:** Here, we compute the ratio of the number of central nodes from the number of nodes in the graph. A node  $u$  is considered as central point if the value of its eccentricity is equal to the effective radius of the graph, i.e.,  $e(u) = rad(G)$ .
11. **Percentage of end points:** It represents the ratio of the number of end points from the total number of nodes of the graph. A node  $u$  is considered as end point if  $deg(u) = 1$ .
12. **Number of distinct eigenvalues:** Any graph  $G$  can be represented by an adjacency matrix  $A$ . As the adjacency matrix  $A$  has a set of eigenvalues, these eigenvalues are not necessarily different. Here, we count the number of distinct eigenvalues of  $A$ .
13. **Spectral radius:** Let  $A$  be the adjacency matrix of the graph  $G$  and  $\lambda_1, \lambda_2, \dots, \lambda_m$  be the set of eigenvalues of  $A$ . The spectral radius of  $G$ , denoted  $\rho(G)$ , represents the largest magnitude eigenvalue, i.e.,  $\rho(G) = \max(|\lambda_i|)$  where  $i \in \{1, \dots, m\}$ .
14. **Second largest eigenvalue:** The value of the second largest eigenvalue of the adjacency matrix of the graph.

15. **Energy:** The energy of an adjacency matrix  $A$  of a graph  $G$  is defined as the squared sum of the eigenvalues of  $A$ . Formally:  $E(G) = \sum_{i=1}^n \lambda_i^2$ .
16. **Neighborhood impurity:** The impurity degree of a node  $u$  belonging to a graph  $G$ , having a label  $L(u)$  and a neighborhood (adjacent nodes)  $N(u)$ , is defined as  $ImpurityDeg(u) = |L(v) : v \in N(u), L(u) \neq L(v)|$ . The neighborhood impurity of a graph  $G$  represents the average impurity degree over all nodes with positive impurity.
17. **Link impurity:** An edge  $\{u, v\}$  is considered to be impure if  $L(u) \neq L(v)$ . The link impurity of a graph  $G$  with  $k$  edges is defined as:  $\frac{|\{u, v\} \in E : L(u) \neq L(v)|}{k}$ .

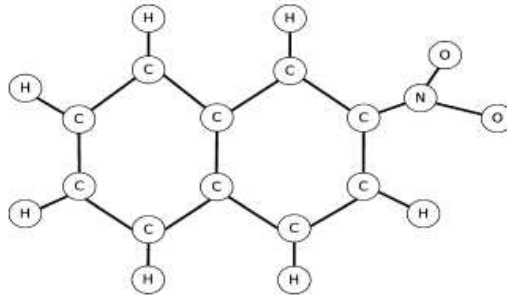


Figure 5.1: An example of a graph of a chemical compound. <sup>1</sup>

**Example** Given the graph in Figure 5.1, the corresponding values of each of the defined attributes are as follows:

- Number of nodes = 20,
- Number of edges = 21,
- Average degree = 2.1,
- Density = 0.11,
- Average clustering coefficient = 0,
- Average effective eccentricity = 5.75,
- Effective diameter = 8,
- Effective radius = 4,

<sup>1</sup>Source: [Li 2012]. Labels of nodes represent the atoms: O=oxygen, H=hydrogens, N=nitrogen, C=carbon.

- Closeness centrality = 0.29,
- Percentage of central nodes = 0.15,
- Percentage of end points = 0.45,
- Number of distinct eigenvalues = 20,
- Spectral radius = 2.56,
- Second largest eigenvalue = 2.15,
- Energy = 42,
- Neighborhood impurity = 1.11,
- Link impurity = 0.48

As efficiency and scalability remain big challenges for graph mining algorithms, the proposed description is unified which helps to overcome both challenges. On one hand, these attributes present an efficient description that is able to reveal hidden topological similarities that exact and approximate structural isomorphism do not consider. On the other hand, considering a fixed number of descriptors guarantee that the encoded vectors would be of a fixed size no matter what the number of graphs in the database is. This makes the approach scalable and computationally efficient in real-world applications. Oppositely, the context-vectors in the the naïve approach are as big as the number of graphs in the database which is usually very high in real-world applications. This can highly affect the scalability and computational consumption of the naïve approach.

### 5.3.3.2 K-Medoids clustering

As previously mentioned, our approach follows a two-step selection framework. First, we discussed the first part of the framework which consists of the description of the data whether by the context-vectors or by the topological description-vectors. Here, we discuss the second part of our selection approach which is the clustering step. We use *k-Medoids* [Kaufman 1987] which is a well known clustering algorithm that is widely used in unsupervised learning [Jain 2010]. It takes as input a set of objects  $\Omega$  and a number of clusters  $k$ , and gives as output the  $k$  clusters' centers (called *medoids*). To do so, k-Medoids uses these definitions:

**Definition 19** (*Pairwise distance between objects*) Given two objects  $O_1$  and  $O_2$  correspondingly described by the vectors  $X$  and  $Y$ , the distance between them, denoted  $d(O_1, O_2)$ , is defined as follows:

$$d(O_1, O_2) = \sum_{i=1}^{|X|} |x_i - y_i|$$

**Definition 20** (*Global distance between objects*) Given a set of objects  $\Omega$ , the total distance between an object  $O$  and all the other ones in  $\Omega$  is defined by:

$$D_O = \sum_{O_i \in \Omega \setminus O} d(O, O_i)$$

**Definition 21** (*Cluster medoid*) An object  $O^*$  is said to be cluster's medoid (the most centrally located object of the cluster), if it has the minimum sum of distances to all the other objects  $O_i$  within the cluster  $C$ . Formally:

$$D_{O^*} = \min_{O_i \in C} (D_{O_i})$$

Using real objects as the clusters' centers makes k-Medoids less sensitive to noise and outliers than many other clustering algorithms. Besides, in k-Medoids, medoids are real data objects. Each medoid represents the most similar object to all the other ones within the same cluster. Thus, medoids can be directly considered as the representative-delegates for all the objects in the same cluster.

The general algorithm of k-Medoids is described in Algorithm 3. First, it starts by randomly selecting  $k$  objects from  $\Omega$  to be the medoids, *i.e.*  $\Omega^*$ . Then, it assigns each non-selected object to the cluster of the nearest medoid. After that, it swaps the  $k$  medoid objects with other non-medoid objects aiming to minimize the overall distance.  $D(\Omega^*)$  is the total distance before the swap and  $D(\Omega'_k)$  is the total distance after the swap. If the cost of the swap ( $C = D(\Omega'_k) - D(\Omega^*)$ ) is strictly negative then the swap is considered as beneficial, otherwise it is ignored. The assignment and swap steps are iteratively performed until no change or until a user-defined maximum number of iteration is reached. Many implementations of k-Medoids have been proposed in the literature. PAM [Kaufman 1987] is a pioneer implementation of k-Medoids. Later, two other implementations have been proposed which are CLARA [Kaufman 1990] and CLARANS [Ng 1994, Ng 2002]. The main difference between these implementations is in the way of performing the swap where in attempt to make the algorithm more scalable to larger amounts of data. In this work, we use CLARANS since it was shown [Ng 2002] that it is an efficient implementation for large-scale data clustering and it gives similar clustering quality to PAM and CLARA.

**Property 3** (*Termination*) There is only a finite number of possible partitionings of the set of objects  $\Omega$  into  $k$  groups. As we are looking for the partitioning that best minimizes the overall distance, we do not go from one partitioning to another only if it improves the clustering. Thus, in each swap

---

**Algorithm 3:** K-MEDOIDS

---

**Data:** Set of objects  $\Omega$ , number of clusters  $k$ , maximum number of iterations  $max_{iter}$

**Result:** Set of medoids  $\Omega^* = \{O_1, O_2, ..., O_k\}$

```

1 begin
2    $\Omega^* \leftarrow \Omega_k$ : start with  $K$  objects randomly selected from  $\Omega$ ;
3   repeat
4     Assign each one of the non-selected objects to the cluster having
       the most similar medoid;
5     Calculate the cost  $C_i = (D(\Omega'_k) - D(\Omega^*))$  for each swap of one
       medoid with another object;
6     if  $C_i < 0$  then
7       |  $\Omega^* \leftarrow \Omega'_k$ ;
8     end
9      $nb_{iter} = nb_{iter} + 1$ ;
10  until (no change) or ( $nb_{iter} \geq max_{iter}$ );
11 end

```

---

*the algorithm must choose a new partitioning. Consequently, after a finite number of iterations, bounded by a user-defined maximum number of iterations, the algorithm will run out of partitionings or no improvement will be observed. Hence, the algorithm terminates.*

### 5.3.3.3 Why k-Medoids and not k-Means?

*K-Means* [MacQueen 1967] is one of the most used algorithms for clustering. We adopt the k-Medoids clustering instead of k-Means because the latter defines the clusters' centers as fictive points. Thus, in order to detect the subgraph delegates, we have to compute the distance between the subgraphs and the center within the same cluster and consider the closest subgraph to the centroid as the representative subgraph delegate. Whereas, the k-Medoids algorithm requires that the clusters' centroids be real points instead of being fictive. Hence, the clusters' medoids are directly considered as the representative subgraph delegates which prevents performing unnecessary computation needed to detect the delegates with k-Means. Besides k-Medoids is less sensitive to noise and outliers.

### 5.3.3.4 The main algorithm

We propose **TRS**, an approach for selecting **T**opological **R**epresentative **S**ubgraphs. The general algorithm of the approach is described in Algorithm 4. **TRS** follows a two steps framework. As previously mentioned, **TRS** assumes that structurally similar subgraphs have similar topological properties. Thus, in the first step of the approach, each subgraph is encoded into a topological description-vector using the previously defined topological attributes. The second step uses the topological description-vectors to select the representative subgraphs. Each representative subgraph is supposed to have the maximal overall similarity to all the other subgraphs it represents. Hence, the topological description-vectors are considered for clustering using k-Medoids. The selected medoids are considered as the topological representative subgraph-delegates.

---

**Algorithm 4:** TRS

---

**Data:** Frequent subgraphs  $\Omega$ , number of representatives  $k$

**Result:** Topological representative subgraphs  $\Omega^* = \{g_1, g_2, \dots, g_k\}$

1 **begin**

2      $M \leftarrow \cup_{i=1}^{|\Omega|} V_i$ : each subgraph  $g \in \Omega$  is encoded into a topological description vector  $V$  using the topological attributes;

3      $\Omega^* \leftarrow \text{K-Medoids}(M, k)$ ;

4 **end**

---

**Property 4** (*Termination*) Since *k-Medoids* terminates, *TRS* terminate too.

## 5.4 Experimental analysis

### 5.4.1 Datasets

To experimentally evaluate our approach, we use different types of graph datasets: protein 3D-structures and chemical compounds. Table 5.1 summarizes the characteristics of the four datasets: dataset,  $|G|$ , Avg. $|V|$  and Avg. $|E|$  correspond respectively to the name of the corresponding protein family or chemical compound dataset, number of graph, average number of nodes, average number of edges in each dataset.

The first two datasets were previously used in [Fei 2010] and [Yan 2008]. Both datasets will be used to evaluate the interestingness of the selected subgraphs. In fact, each dataset is composed of two groups of protein 3D-structures equally divided between positive and negative samples. Positive



Table 5.1: Benchmark datasets

Dataset	$ G $	Avg. $ V $	Avg. $ E $
G-proteins	66	246	971
C1 set domains	76	238	928
Enzymes	664	358	910
AIDS antiviral screen	43850	28	30

proteins are sampled from a selected protein family, namely G-proteins and C1 set domains, whereas negative proteins are randomly sampled from the Protein Data Bank [Berman 2000]. G-proteins are also known as guanine nucleotide-binding proteins. These proteins are mainly involved in transmitting chemical signals originating from outside a cell into the inside of it. They regulate metabolic enzymes, ion channels, transporter, and other parts of the cell machinery, controlling transcription, motility, contractility, and secretion, which in turn regulate diverse systemic functions such as embryonic development, learning and memory, and homeostasis.

The C1 set domains composing the second dataset are immunoglobulin-like domains, similar in structure and sequence. They resemble the antibody constant domains. They are mostly found in molecules involved in the immune system, in the major histocompatibility complex class I and II complex molecules, and in various T-cell receptors. The two other datasets are used to evaluate the runtime and the distribution of subgraphs according to their sizes. The dataset of Enzymes, previously used in [Dobson 2003] and [Thoma 2010], is composed of 664 proteins. Enzymes act as biological catalysts. They are large biological molecules responsible for the thousands of chemical interconversions that sustain life. The last dataset shows a set of antiviral screen data (AIDS). It contains the activity test information of 43850 chemical compounds. This dataset was previously used in many studies such as [Chen 2008] and is publicly available on the website of the Developmental Therapeutics Program.<sup>2</sup>

### 5.4.2 Protocol and settings

**Graph building:** For chemical compounds, each atom is represented by a node and labeled with the atom type (Hydrogen (H), Carbon (C), etc.). An edge exists between two nodes if there exists a chemical bond between their corresponding atoms. For protein 3D-structures, each protein is parsed into a graph of amino acids using the main atom ( $C_\alpha$ ) method (see section 2.3.4.2). In the literature, many methods use this method with usually  $\delta \geq 7\text{\AA}$  on the

<sup>2</sup>[http://dtp.nci.nih.gov/docs/aids/aids\\_data.html](http://dtp.nci.nih.gov/docs/aids/aids_data.html)

argument that  $C_\alpha$  atoms define the overall shape of the protein conformation [Huan 2005]. In our experiments, we use  $\delta = 7\text{\AA}$ .

**Frequent subgraph mining:** We use the state-of-the-art method of frequent subgraph discovery GSPAN [Yan 2002] to find the frequent subgraphs in each dataset. We tried different minimum frequency threshold in order to obtain a reasonable number of frequent subgraphs from each dataset. The retained minimum frequency threshold are 30% for G-proteins and C1 set domains, 10% for Enzymes, and 5% for AIDS antiviral screen dataset. Table 5.2 shows the number of frequent subgraphs obtained from each dataset.

**Representative subgraph selection:** Both selection frameworks, *i.e.*, the

Table 5.2: Number of frequent subgraphs ( $\Omega$ ) extracted from each dataset

Dataset	$ \Omega $
G-proteins	114792
C1 set domains	258371
Enzymes	253404
Sida	6749

naïve approach and TRS, were implemented in R.

**Subgraph encoding:** To measure the quality of subgraphs, each one of them is encoded into a binary vector by denoting 1 or 0, the presence or the absence of the subgraph in each graph in the dataset. The quality of the selected subgraphs is measured over their encoding vectors.

## 5.5 Results and discussion

### 5.5.1 Empirical results

As previously mentioned, we first evaluate our approach over the classification datasets G-proteins and C1 set domains. We measure the quality of the selected subgraphs using the information gain which is one of the most popular interestingness measures in data mining. Given a set of training examples  $\Omega$  and an attribute  $att$ . The information gain of  $att$  is computed using the following formulas:

$$InformationGain(\Omega, att) = Entropy(\Omega) - Entropy(\Omega|att)$$

where  $Entropy(\Omega)$  is calculated as follows:

$$Entropy(\Omega) = -\sum_{i=1}^{|\Omega|} p(x_i) \log p(x_i)$$

where  $p(x_i)$  is the probability of getting the  $x_i$  value when randomly selecting an example from the set.

The information gain is measured over all the frequent subgraphs then over the subgraphs selected by TRS and those selected by the naïve approach using different number of representatives. The information gain value obtained over all the frequent subgraphs is considered as standard value for comparison. Table 5.3 shows the obtained results.

Table 5.3: Comparison of average information gain of the topological representative subgraphs (TRS) with those selected by the naïve approach (NA) and the initial set of all frequent subgraphs (FSG).

	G-proteins		C1 set domains	
FSG	0.216		0.148	
# representatives	NA	TRS	NA	TRS
50	0.104	0.324	0.068	0.254
100	0.092	0.342	0.061	0.285
200	0.096	0.343	0.044	0.273
300	0.097	0.347	0.058	0.267
400	0.094	0.339	0.051	0.276
500	0.090	0.348	0.052	0.269
600	0.096	0.340	0.054	0.267
700	0.097	0.343	0.055	0.272
800	0.098	0.352	0.054	0.274
900	0.094	0.358	0.054	0.276
1000	0.094	0.353	0.056	0.276
<b>Average</b>	$0.095^{+0.008}_{-0.005}$	$0.344^{+0.013}_{-0.020}$	$0.055^{+0.012}_{-0.011}$	$0.271^{+0.013}_{-0.017}$

Table 5.3 shows that TRS is able to select a subset of subgraphs that are more informative than either the initial frequent ones or those selected by the naïve approach. Whereas, the quality of the subsets of representative subgraphs selected by the naïve approach did not even reach the information gain value of the whole set of frequent subgraphs. Both previous interpretations goes with all the used numbers of representatives. This proves the reliability of our selection approach and shows that using the topological attributes for description is more efficient than using the occurrence information. It enables k-Medoids to better detects similarities between subgraphs and thus to select a subset of representative subgraphs that are most informative.

It is also worth mentioning that the topological attributes used in TRS are not limited to the ones mentioned in this chapter. They can be extended by removing or adding other attributes depending on the data and the application

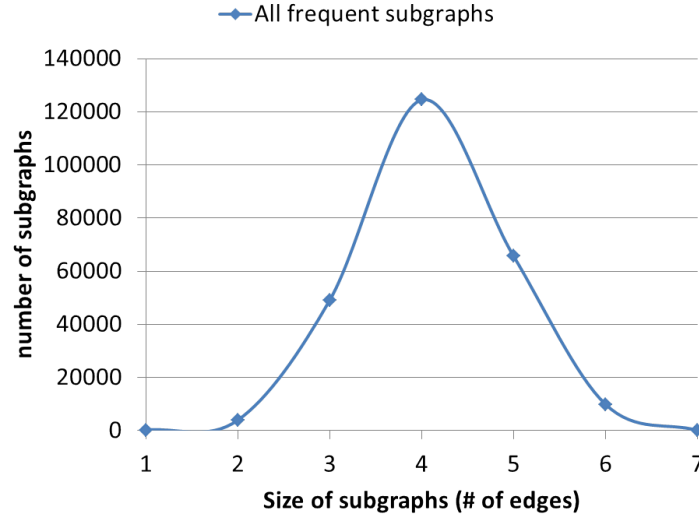


Figure 5.2: Distribution of subgraphs by size for the Enzymes dataset.

goals. For instance, for graphs containing loops, it could be interesting to consider the number of loops in each subgraph as feature.

### 5.5.2 Size-based distribution of patterns

In this section, we study the distribution of subgraphs based on their size (number of edges). We try to check which sizes of subgraphs are more concerned by the selection. The Figures 5.2 and 5.3 draw the distribution of the original set of frequent subgraphs over Enzymes and AIDS antiviral screen datasets. For both datasets, we notice a high concentration of the number of frequent subgraphs in the center especially with the Enzymes dataset, ranging from three to five for the latter, and from four to nine for the AIDS antiviral screen dataset. These concentration zones presents high level of redundancy and must be the most concerned by the selection.

Figures 5.4 and 5.5 draw the distribution of the topological representative subgraphs with different number of clusters  $k$ . The downward tendency of TRS using lower values of  $k$  and with respect to the original set of frequent subgraphs is very clear. In fact, TRS leans towards cutting off the peaks and flattening the curves with lower value of  $k$ . Another interesting observation is that the curves are flattened in the regions of small subgraphs as well as in those of large subgraphs. This demonstrates the effectiveness of TRS on both small and large subgraphs.

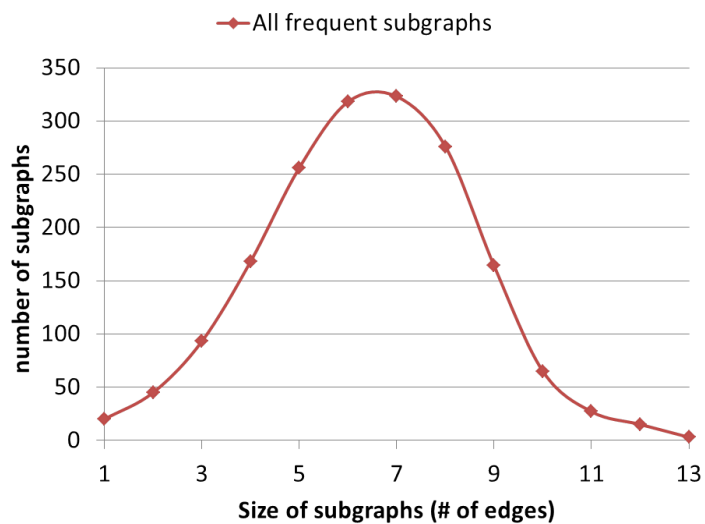


Figure 5.3: Distribution of subgraphs by size for the AIDS antiviral screen dataset.

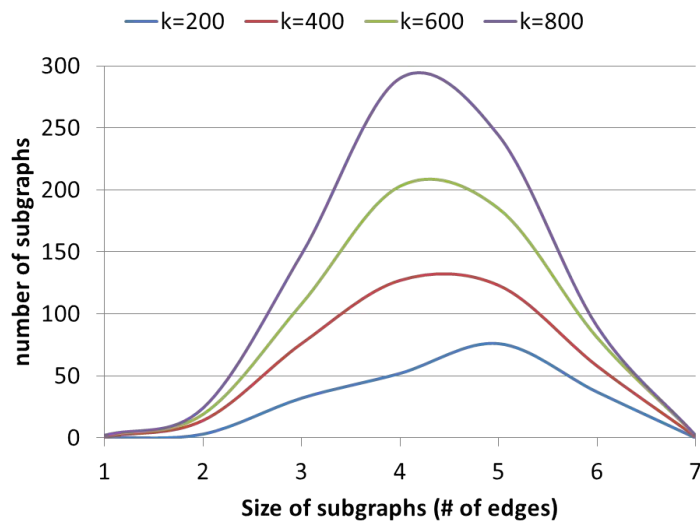


Figure 5.4: Distribution of subgraphs by size for the Enzymes dataset using different values of  $k$  (number of representatives).

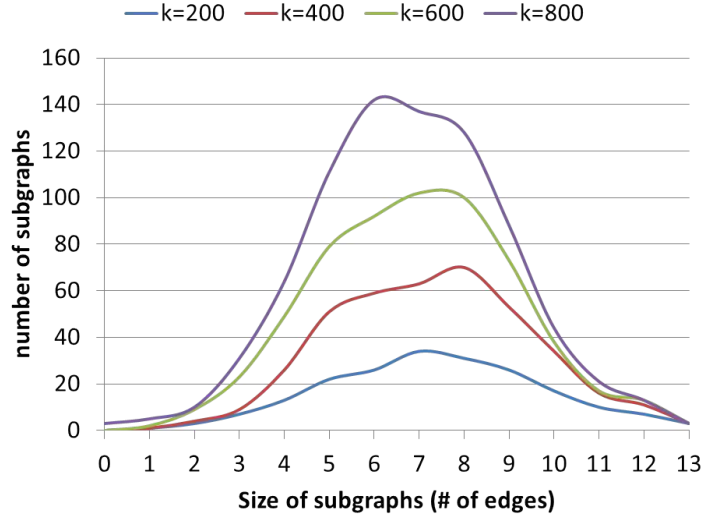


Figure 5.5: Distribution of subgraphs by size for the AIDS antiviral screen dataset using different values of  $k$  (number of representatives).

### 5.5.3 Runtime analysis

In this section, we study the runtime of our algorithm compared to that of the naïve approach on three levels: in terms of variation of number of clusters, numbers of frequent subgraphs, and number of graphs. It is worth mentioning that here we only compare the clustering runtime and we omit the time of the encoding of subgraphs since it does not change along the experiments and only counts few seconds. Besides, it depends on the selected attributes for TRS.

#### 5.5.3.1 Scalability to higher number of clusters

We study the effect of varying the number of clusters  $k$  on the runtime of clustering for both TRS and the naïve approach. We select the representative subgraphs among the frequent ones previously extracted from the AIDS antiviral screen dataset. Figure 5.6 illustrates the evolution of runtime using different values of  $k$  (number of clusters) ranging from 200 to 800 with a step-size of 200.

Figure 5.6 shows a huge difference in execution time between the two approaches. In fact, for 200 clusters, the naïve approach consumes more than one and half hour to finish the clustering, whereas TRS needed only few seconds. This difference becomes much bigger with higher values of  $k$ . As the number of clusters increases, the execution time of the naïve approach exponentially increases as well. Yet, the clustering time in TRS does not

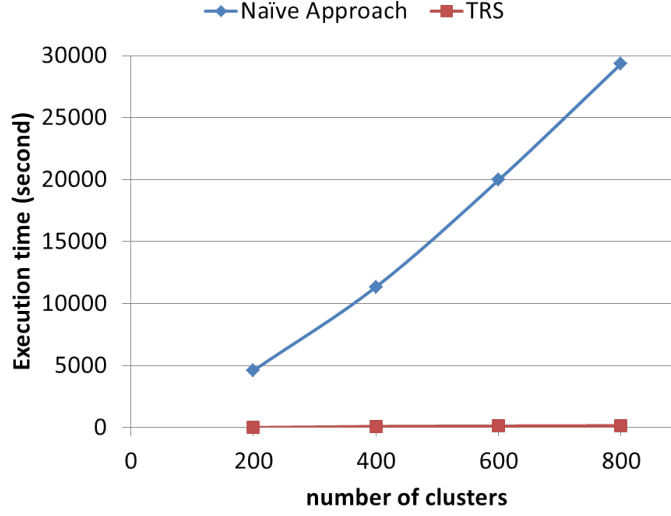


Figure 5.6: Runtime of clustering for TRS and the naïve approach with different number of clusters ( $k$ ).

increase significantly and almost stays steady with higher values of  $k$ . Since the clustering is combinatorial and considers each possible pair of subgraphs for comparison, the smaller the description of the subgraphs is, the faster the clustering would be. Consequently, the huge gain in execution time is basically due to the small and fixed size of the topological description-vectors used in TRS compared to the context description-vectors in the naïve approach.

### 5.5.3.2 Scalability to higher number of subgraphs

Here, we study the effect of varying the number of frequent subgraphs on clustering runtime for both TRS and the naïve approach. We select the representative subgraphs among different sets of frequent subgraphs ranging from 10000 to 100000 with a step size of 10000. The input subgraphs were randomly selected among the frequent subgraphs previously extracted from the C1 set domains dataset. The Figures 5.7 and 5.8 illustrate the evolution of runtime with higher number of subgraphs, respectively for 100 and 500 clusters.

As shown in the Figures, TRS takes only few seconds to select the representative subgraphs, whereas, the naïve approach takes clearly much more time. Increasing the number of subgraphs does not affect the runtime of TRS as much as it does with the naïve approach. This shows that TRS is more scalable than the naïve approach to higher numbers of subgraphs.

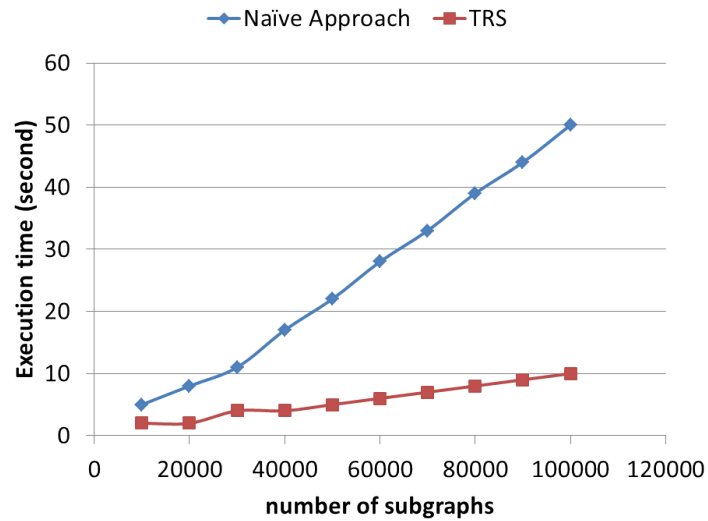


Figure 5.7: Runtime of clustering for TRS and the naïve approach to select 100 representatives among different number of subgraphs.

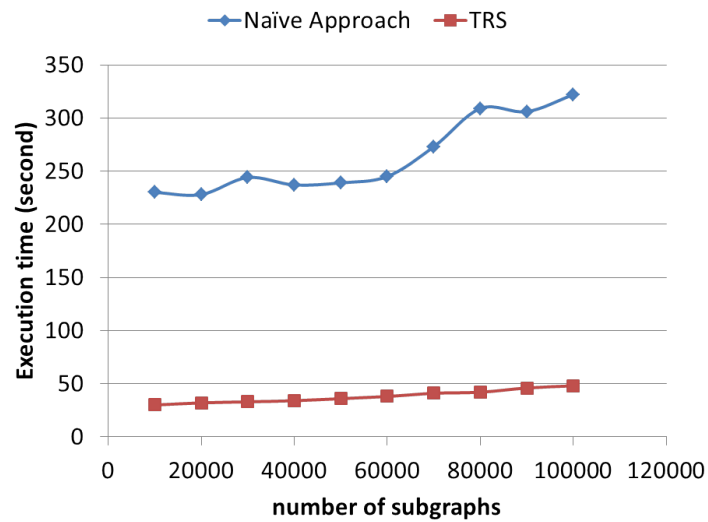


Figure 5.8: Runtime of clustering for TRS and the naïve approach to select 500 representatives among different number of subgraphs.



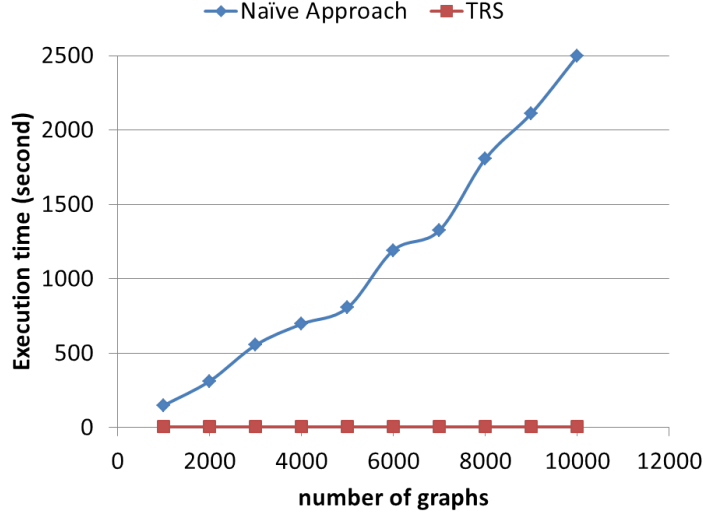


Figure 5.9: Runtime of clustering for TRS and the naïve approach to select 100 representatives among 10000 subgraphs with variation of the number of graphs.

### 5.5.3.3 Scalability to higher number of graphs

In real-world applications, the size of graph databases is usually very high. We study the effect of varying the number of graphs on the runtime of both TRS and the naïve approach. We fix the number of subgraphs to 10000, and we synthetically manipulate the list of occurrences of each frequent subgraph by replacing it with a random list of random occurrences between 0 and a considered number of graphs. The considered numbers of graphs in the occurrence lists range from 1000 to 10000, with a step size of 1000. Figures 5.9 and 5.10 illustrate the evolution of runtime with higher number of graphs, respectively for 100 and 500 clusters.

As the naïve approach uses the occurrence lists of subgraphs to construct the context description-vectors. Thus, the size of each context-vector is equal to the number of graphs in the database. Consequently, the runtime of the naïve approach is highly affected by the increasing of the number of graphs in the database. Both Figures 5.9 and 5.10 show that the runtime of the naïve approach increases exponentially with higher numbers of graphs. Whereas, the runtime of TRS corresponds only to few seconds and remains stable no matter what the size of the database is. This shows that TRS is scalable and more robust in real world-applications that usually deals with large amounts of data.

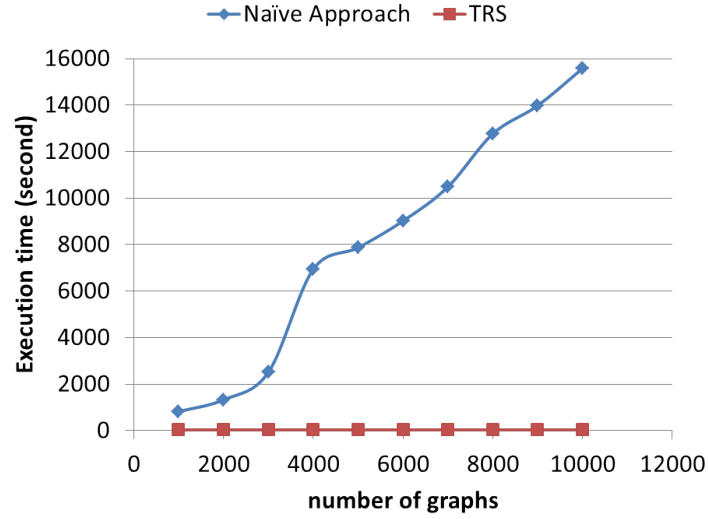


Figure 5.10: Runtime of clustering for TRS and the naïve approach to select 500 representatives among 10000 subgraphs with variation of the number of graphs.

## 5.6 Conclusion

In this chapter, we proposed TRS, a novel approach that mines a subset of topological representative subgraphs among the frequent ones. Instead of exact and approximate structural similarity, TRS follows a more meaningful selection strategy that helps on both selecting a subset of topologically non-redundant and informative subgraph-delegates. TRS is also able to detect hidden topological similarities between subgraphs that are ignored by current selection approaches, and can be easily extended using any user-defined attributes. Empirical studies on real and synthetic graph datasets showed that our approach is fast and scalable. Besides graph databases, it can handle other cases such as the problem of subgraph selection in a single large graph.

In many application, the user may not be able to define a specific number of clusters. A promising future direction could be to remove the  $k$  constraint. This is further discussed in Section 6.4.2.2.



# Conclusion and future works

---

## Contents

---

<b>6.1</b>	<b>Aims</b>	<b>109</b>
<b>6.2</b>	<b>Summary of contributions</b>	<b>110</b>
6.2.1	UNSUBPATT	110
6.2.2	TRS	110
<b>6.3</b>	<b>Discussion</b>	<b>111</b>
<b>6.4</b>	<b>Ongoing works and prospects</b>	<b>113</b>
6.4.1	UNSUBPATT extensions	113
6.4.2	TRS extensions	114

---

## 6.1 Aims

In this chapter, we conclude the thesis by summarizing the proposed contributions and highlighting some ongoing works for both UNSUBPATT and TRS.

## 6.2 Summary of contributions

In this thesis, we proposed two feature selection approaches for subgraphs. Here, we recall both approaches as well as the main results and conclusions.

### 6.2.1 UNSUBPATT

The first approach we proposed is termed UNSUBPATT. It aims to selecting a subset of representative subgraphs among frequent ones. The selected subgraphs are termed *representative **un**substituted **p**atterns*. Unlike existing subgraph selection approaches where the *prior* domain knowledge is often ignored, UNSUBPATT incorporates matrices that quantify the similarities between nodes labels. UNSUBPATT uses similarity scores of the matrix to detect the overall similarity between pairs of subgraphs. Graphs representing protein structures are an immediate application example due to the availability of amino acids substitution matrices. However, UNSUBPATT can be used in any other application context whenever it is possible to define a matrix that quantifies similarities between the nodes' labels. UNSUBPATT is unsupervised, thus, it can be used in any subgraph-based task.

Experimental evaluation of UNSUBPATT was performed by classifying a set of protein structure datasets. Results showed that UNSUBPATT is able to select a small yet representative and informative subset of subgraphs among the frequent ones. Moreover, UNSUBPATT outperformed many other subgraph selection approaches in classifying the considered protein structure datasets. It even reached full accuracy with one dataset. This shows that UNSUBPATT is a very competitive and promising approach, and that using the substitution between amino acids allows it to select a very informative subset of subgraphs.

### 6.2.2 TRS

Similarity in UNSUBPATT is purely semantic as the similarity between a pair of isomorphic subgraphs depends on how similar their labels are. We also introduced another subgraph selection approach, we term TRS (**T**opological **R**epresentative **S**ubgraphs). Redundancy in UNSUBPATT is based on semantic similarity, while redundancy in TRS is based on structural similarity. Existing subgraph selection approaches that are based on structural similarity are either exact or approximate. We discussed, in 5.2, how current exact and approximate structural similarity approaches are less efficient in many real-world applications. Unlike these approaches, TRS follows a more meaningful selection by considering the overall structural similarity between subgraphs

through a set of topological descriptors. This makes it easily extendable with any user-specified descriptors depending on the application and the sought information.

Experimental evaluation of TRS was performed mainly on protein structure datasets but also on a chemical compound dataset. Results showed that TRS is able to select a set of topologically non-redundant and informative subgraph-delegates. In addition, it considers hidden topological similarities between subgraphs (density, diameter, clustering coefficient, *etc*) that are ignored by current selection approaches. Moreover, TRS is extendable and unsupervised, thus it can be used in any subgraph-based task. It is also worth noting that the application domain of TRS is not limited to protein 3D-structures or to biological data but it can also be used with any graph data.

## 6.3 Discussion

We resume the discussion previously reported in 3.7. As previously discussed, many subgraph selection approaches are currently available. It is difficult to compare them, in general, since the majority of them were originally designed to resolve a particular issue. The choice of an appropriate selection method highly depends on the users preferences and the application constraints. In Table 6.1, we list all the subgraph selection approaches that have been investigated along Chapter 3 and we state their characteristics according to a set of descriptors. In addition to what was reported in Table 3.1, Table 6.1 lists our proposed approaches, UNSUBPATT and TRS. It also contains an additional descriptor which indicates whether the selection approach considers the prior domain knowledge in the selection or not. It is possible to consider similarity functions and measures, that are defined by the user, as domain knowledge like in Redundancy aware top-k, SkyGraph, Mips or TRS. However, here we refer to prior domain knowledge as specific external data or information from the application domain that a method exploits during the selection, as in UNSUBPATT which uses the substitution matrices that are already defined by domain experts.

Table 6.1: Characteristics of Subgraph selection approaches according to different discription criterions.

<i>Subgraph selection approach</i>	<i>Descriptor</i>			
	Prior domain knowledge	Post- processing	Learning- task de- pendent	Selected subgraphs
TGP	No	No	No	Top-k frequent closed
Redundancy aware top-k	No	Yes	No	Top-k frequent significant&non-redundant
RP-FP	No	Yes	No	Frequent closed representatives
RP-GD	No	No	No	Frequent closed representatives
RING	No	No	No	Frequent representatives
ORIGAMI	No	Yes	No	$\alpha$ -orthogonal $\beta$ -representative
Output space sampling	No	No	No	Sample of frequent
MCSs sample	No	No	No	Maximum common subgraphs
Smoothing-clustering	No	Yes	No	Approximate structural representatives
D&D	No	No	Yes	Diverse discriminative
GAIA, CORK, GraphSig, LEAP, LPGBCMP, COM, gBoost, gPLS	No	No	Yes	Discriminative
MIPs	No	No	No	Most informative closed
SkyGraph	No	No	No	Undominated
Ant-motifs	No	No	No	Ant-like shape
UNSUBPATT	Yes	Yes	No	Representative unsubstituted
TRS	No	Yes	No	Topological representative

## 6.4 Ongoing works and prospects

Like any algorithm, the proposed selection approaches have their limitations. In the following, we discuss some of the major limitations and we propose possible extensions to enhance them.

### 6.4.1 UNSUBPATT extensions

#### 6.4.1.1 Parallel UNSUBPATT

Even though UNSUBPATT scales well with higher numbers of subgraphs, the problem still of high complexity due to the combinatorial test of substitution between subgraphs. In real-world applications, the number of subgraphs can be exponential. It would be interesting to make UNSUBPATT runs faster to be able to deal with exponential numbers of subgraphs and to be more efficient in real-world applications. A possible way to make UNSUBPATT run faster is parallelization. UNSUBPATT tests separately the substitution among groups of subgraphs having the same size and order. Hence, an easy way to parallelize it is to test the substitution in groups in parallel threads or processors or even machines as in a grid or a cloud environment.

#### 6.4.1.2 Approximate early termination

Another possible way to make UNSUBPATT runs faster and more efficiently is to integrate the selection in the subgraph extraction process through an early termination condition. Introducing such condition is very difficult in either breadth or depth first search approach, as there is no guarantee that the resulting set of selected subgraphs is the optimal representative set. Moreover, the resulting set may only cover a small portion of the search space since the search would not be complete in many branches of the search tree. Indeed, in a breadth first search approach, the output subgraphs would be only up to some levels of the search tree, and in a depth first search, the selected subgraphs would cover other branches of the search tree only to a certain levels. In both cases, there is no guarantee that the cut branches do contain only irrelevant and redundant subgraphs. Thus, many representative subgraph candidates may be lost.

A possible way to perform UNSUBPATT selection during the extraction of subgraphs is through approximation. Although this do not guarantee selecting the optimal set of representatives, we claim that this may provide a near optimal solution. For any frequent subgraph  $P_1$ , if it substitutes another subgraph  $P_2$  and a child node  $P_{11}$  of  $P_1$  (in the search tree) also substitutes a child node  $P_{21}$  of  $P_2$ , then the growing stops from  $P_{21}$ . Otherwise, the growing



continues until the same condition is verified or no other frequent subgraph is discovered. The frequent subgraphs that have not been substituted represent the set of representative unsubstituted patterns.

## 6.4.2 TRS extensions

### 6.4.2.1 Parallel TRS

In real-world applications, the number of subgraphs can be exponential. Although TRS scales well with higher numbers of subgraphs, making TRS runs faster would be very interesting for real-world applications. Since TRS is composed of two steps, parallelization should cover both of them. The first step consists in computing the corresponding values of the topological attributes for each subgraph. This step can be parallelized easily in two ways: the first way is to compute the values of each attribute in parallel processes such that each process deals with one attribute for all subgraphs. The second way, is to divide the subgraph set in different groups then to compute the values of all attributes for each group in parallel processes such that each process computes the values of all attributes for a single group. The second step of TRS consists in clustering subgraphs based on their description-vectors into groups using k-Medoids. In [Gamblin 2010], authors proposed *CAPEK*, a massively scalable parallel version of k-Medoids clustering algorithm. TRS can use *CAPEK* in the second step. Hence, TRS can be fully parallelized.

### 6.4.2.2 Removing the $k$ constraint

In many applications, the user may not be able to define a specific number of clusters. An interesting extension of TRS is to remove the  $k$  constraint. This can be performed using a clustering algorithm that do not require specifying the number of clusters. For instance, *CAPEK* [Gamblin 2010] can determine the value of  $k$  automatically and thus it eliminates the need to specify the number of clusters in advance. This can also be performed using Medoidshift [Sheikh 2007] which is a non-parametric partitioning algorithm that automatically computes the number of clusters. Many other clustering techniques also offer this possibility such as hierarchical and density-based clustering.

# Bioinformatics data formats

---

## A.1 PDB format

A PDB file is a textual format describing the position of atoms in a molecule in the 3D-space. To reduce the size of PDB files, the hydrogen atoms are omitted from the description files of macromolecules. Even for small molecules, the double bonds are rarely present. An example of a PDB file is in Figure A.1. The file describes the coordinates of the atoms that are part of the protein. For example, the first ATOM line above describes the alpha-N atom of the first residue of peptide chain A, which is a proline residue, the first three float numbers are its  $x$ ,  $y$  and  $z$  coordinates and are in units of Angstroms. The next three columns are respectively the occupancy, temperature factor, and the element name.

```

HEADER    EXTRACELLULAR MATRIX                22-JAN-98   1A3I
TITLE     X-RAY CRYSTALLOGRAPHIC DETERMINATION OF A COLLAGEN-LIKE
TITLE     2 PEPTIDE WITH THE REPEATING SEQUENCE (PRO-PRO-GLY)
...
EXPDTA    X-RAY DIFFRACTION
AUTHOR    R.Z.KRAMER,L.VITAGLIANO,J.BELLA,R.BERISIO,L.MAZZARELLA,
AUTHOR    2 B.BRODSKY,A.ZAGARI,H.M.BERMAN
...
REMARK 350 BIOMOLECULE: 1
REMARK 350 APPLY THE FOLLOWING TO CHAINS: A, B, C
REMARK 350   BIOMT1   1   1.000000   0.000000   0.000000           0.00000
REMARK 350   BIOMT2   1   0.000000   1.000000   0.000000           0.00000
...
SEQRES    1 A      9  PRO PRO GLY PRO PRO GLY PRO PRO GLY
SEQRES    1 B      6  PRO PRO GLY PRO PRO GLY
SEQRES    1 C      6  PRO PRO GLY PRO PRO GLY
...
ATOM      1  N      PRO A  1           8.316  21.206  21.530  1.00 17.44      N
ATOM      2  CA      PRO A  1           7.608  20.729  20.336  1.00 17.44      C
ATOM      3  C       PRO A  1           8.487  20.707  19.092  1.00 17.44      C
ATOM      4  O       PRO A  1           9.466  21.457  19.005  1.00 17.44      O
ATOM      5  CB      PRO A  1           6.460  21.723  20.211  1.00 22.26      C
...

```

Figure A.1: PDB format.

## A.2 FASTA format

FASTA format is used to represent biological sequences, such as protein primary structures, in textual files. In a FASTA file, each biological sequence is represented by a string of characters where each character represents a nucleotide or an amino acid and are ordered as the latter appear in their sequence. Different sequences are separated by a greater-than (" $>$ ") symbol. The line containing the (" $>$ ") symbol are also used for descriptions such as the name of the sequence. Figure A.2 shows an example of a FASTA file.

```
>ExTopoDBID:1[Uniprot_AC:O04714]
MSAVLTAGGGLTAGDRSIIITAINTGASSLSFVGSFIVLCYCLFKELRKFSFKLVFYAL
SDMLCSFFLIVGDPKGFICYAQGYTTHFFCVASFLWTTTIAFTLHRTVVKHKTDVEDLE
AMFHLVVMGTSLVVTVIRSFNMNHSGLPWCWTQTGLKGAHVHFLTFFYAPLWGAILYNGF
TYFQVIRMLRNARRMAVGMSDRVDQFDNRAELKVLNRWGYYPILILIGSWAFGTINRIHDF
IEPGHKIFWLSVLDVGTAAALMGLFNSIAYGFNSSVRRRAIHERLELFLPERLYRWLPSNFR
PKNHLILHQQQQRSEMVS LKTEDQQ
>ExTopoDBID:10[Uniprot_AC:O34653]
MTEQTIAHKQKQLTKQVAAFAQPETKNSLIQLLNTFIPFFGLWFLAYLSLDVSYLLTLAL
TVIAAGFLTRIFIIFHDCCHQSFFKQKRYNHILGFLTGVLTLPYQLWQHSHSIHHATSS
NLDKRGTDIWMMLTVNEYKAASRRTKLAYRLYRNPFIMFILGPIYVFLITNRFNKKGARR
KERVNTYLTNLAIVALAAACCLIFGWQSFLLVQGPFIPLISGSIGVWLFYVQHTFEDSYFE
ADENWSYVQAAVEGSSFYKLPKLLQMLTGNIGYHHVHHLSPKVPNYKLEVAHEHHEPLKN
VPTITLKTSLQSLAFRLWDEDNKQFVSFRAIKHIPVSLPPDSPEKQKLRKNA
>ExTopoDBID:100[Uniprot_AC:P07340]
MARGKAKEEGSWKKFIWNSEKKEFLGRTGGSWFKILLFYVIFYGCLAGIFIGTIQVMLLT
ISELKPTYQDRVAPPGLTQIPQIQKTEISFRPNPKSYEAYVLNIIIRFLEKYKDSAQKDD
MIFEDCGSMPSPEPKERGEFNHERGERKVCRFKLDWLGNCSGLNDESYGYKEGKPCIIIKL
NRVLGFKPKPPKNESLETYPLTMKYNPNVLPVQCTGKRDEDDKDKVGNIEYFGMGGFYGF
LQYYPYQKLLQPKYLQPLLAVQFTNLTLDTIEIRIECKAYGENIGYSEKDRFQGRFDVKI
EVKS
>ExTopoDBID:1000[Uniprot_AC:P67143]
MNEIISAALLILIMDPLGNLPIMSVLKHTEPKRRRAIMVRELLIALLVMLVFLFAGEK
ILAFSLRAETVSISGGIILFLIAIKMIFPSASGNSSGLPAGEEPFIVPLAIPLVAGPTI
LATLMLLSHQYPNQMGHLVIALLLAWGGTFVILLQSSFLRLLGEKGVNALERLMGLILV
MMATQMFLDGIWMWMMKG
```

Figure A.2: FASTA format.

# Protein Graph Repository

---

## B.1 Description

Protein Graph Repository (PGR) is an online repository mainly dedicated to graphs representing protein 3D-structures. The core of this online repository is developed using both JAVA and PHP as a programming languages and MySQL as a database management system. PGR was deployed using the latest web technologies and respecting the web standardization specifications.

## B.2 How to use PGR ?

The general operation schema of PGR is as follow :

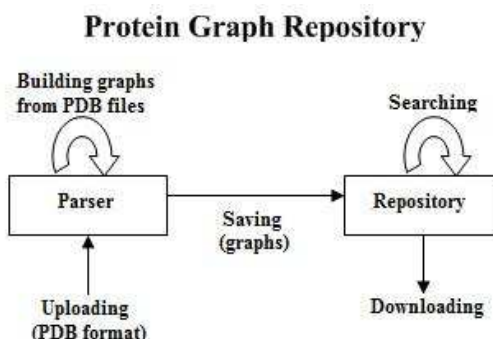
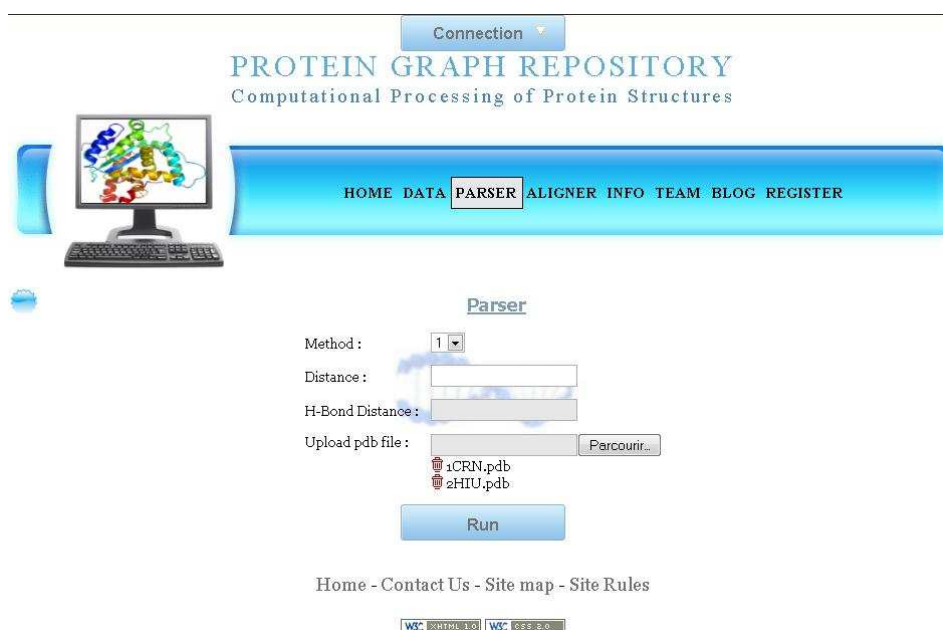


Figure B.1: PGR general schema.

### B.2.1 Parser

The parser tool (see Figure B.2) allows the transformation of PDB files [Berman 2000] (see Section A.1) into graphs (see Figure B.3). Many graph formats can be generated enabling the use of panoply of existing tools such as Biolayout [Theocharidis 2009], Network Workbench [Börner 2010], GraphClust [Recupero 2008], *tec*.

Several methods of graph construction are supported. The use of the parser is very simple:



The screenshot shows the Protein Graph Repository website. At the top, there is a navigation bar with a 'Connection' dropdown menu and a main menu with links: HOME, DATA, **PARSER**, ALIGNER, INFO, TEAM, BLOG, and REGISTER. The 'PARSER' link is highlighted. Below the navigation bar, there is a section titled 'Parser' with a 'Method' dropdown menu set to '1'. There are input fields for 'Distance' and 'H-Bond Distance'. Below these, there is an 'Upload pdb file' section with a file list showing '1CRN.pdb' and '2HIU.pdb'. A 'Parcourir...' button is next to the file list. A 'Run' button is at the bottom of the form. At the very bottom of the page, there are links for 'Home - Contact Us - Site map - Site Rules' and a version indicator 'WS2.0.0'.

Figure B.2: Parser.

- The user upload his set of PDB files
- Specify : the graph construction method, the appropriate parameters values, and the output format
- Run the parser

A more detailed description is reported in the site.

### B.2.2 Repository

The repository (see Figure B.4) represents a protein graph data bank that is freely available online. It is coupled with a filtering tool allowing the selection and targeting of a specific set of protein graphs. The repository is fed each time the parser is run. A download option is enabled making the existing protein graphs available for any further purpose.

```

#url file:/D:/PhD_Project/Development/ProGraMX/ProGraMX/./d1/1J8U.pdb
#graph_building_method BasedOnAllAtoms 3.6Å
#vertices
V 0
P 1
W 2
F 3
P 4
R 5
T 6
.....
V 305
L 306
#edges
V 0 P 1
P 1 W 2
P 1 F 3
P 1 G 194
W 2 F 3
W 2 P 4
W 2 R 12
F 3 P 4
F 3 R 5
F 3 G 194
P 4 R 5
P 4 E 9
P 4 F 142
.....

```

Figure B.3: PGR file.

Connection ▾

**PROTEIN GRAPH REPOSITORY**  
Computational Processing of Protein Structures

UNDER CONSTRUCTION

HOME DATA PARSER INFO TEAM BLOG REGISTER

Data :

Id pdb Method Distance1 Distance2 Distance3 Format

Id pdb	Method	Distance1	Distance2	Distance3	Format
1J8U	BasedOnCAlpha	1.0			gcl
1J8U	BasedOnCAlpha	1.0			nwb
1J8U	BasedOnCAlpha	1.0			sds
1J8U	BasedOnCAlpha	1.0			ing
1J8U	BasedOnCAlpha	1.0			layout
1J8U	BasedOnCAlpha	2.0			pgr

Page 1 of 1 10 View 1 - 6 of 6

download

Home - Contact Us - Site map - Site Rules

WS: 800ML10 WS: 800P20

Figure B.4: Data



# Bibliography

- [Abu-Khzam 2007] F.N. Abu-Khzam, N.F. Samatova, M.A. Rizk and M.A. Langston. *The Maximum Common Subgraph Problem: Faster Solutions via Vertex Cover*. In IEEE/ACS International Conference on Computer Systems and Applications, 2007., pages 367–373, 2007. (Cited on pages [48](#), [49](#) and [88](#).)
- [Altschul 1990] S. Altschul, W. Gish, W. Miller, E. Myers and D. Lipman. *Basic local alignment search tool*. Journal of Molecular Biology, vol. 215, pages 403–410, 1990. (Cited on pages [75](#) and [83](#).)
- [Anchuri 2013] Pranay Anchuri, Mohammed J. Zaki, Omer Barkol, Shahar Golan and Moshe Shamy. *Approximate graph mining with label costs*. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '13, pages 518–526. ACM, 2013. (Cited on pages [50](#) and [64](#).)
- [Andreeva 2008] Antonina Andreeva, Dave Howorth, John-Marc Chandonia, Steven E. Brenner, Tim J. P. Hubbard, Cyrus Chothia and Alexey G. Murzin. *Data growth and its impact on the SCOP database: new developments*. Nucleic Acids Research, vol. 36, no. 1, pages D419–D425, 2008. (Cited on pages [62](#) and [73](#).)
- [Ankerst 1999] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel and Jörg Sander. *OPTICS: ordering points to identify the clustering structure*. In Proceedings of the 1999 ACM SIGMOD international conference on Management of data, SIGMOD '99, pages 49–60. ACM, 1999. (Cited on page [17](#).)
- [Bartoli 2007] L. Bartoli, P. Fariselli and R. Casadio. *The effect of backbone on the small-world properties of protein contact maps*. Physical Biology, vol. 4, no. 4, pages L1+, 2007. (Cited on page [62](#).)
- [Berman 2000] Helen M. Berman, John D. Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov and Philip E. Bourne. *The Protein Data Bank*. Nucleic Acids Research, vol. 28, no. 1, pages 235–242, 2000. (Cited on pages [27](#), [28](#), [62](#), [73](#), [98](#) and [117](#).)
- [Bi 2003] Jinbo Bi and Vladimir Vapnik. *Learning with Rigorous Support Vector Machines*. In COLT, pages 243–257, 2003. (Cited on page [16](#).)



- [Borgelt 2002] Christian Borgelt and Michael R. Berthold. *Mining Molecular Fragments: Finding Relevant Substructures of Molecules*. IEEE International Conference on Data Mining (ICDM), vol. 0, page 51, 2002. (Cited on page 40.)
- [Börner 2010] Katy Börner, Weixia Huang, Micah Linnemeier, Russell J. Duhon, Patrick Phillips, Nianli Ma, Angela Zoss, Hanning Guo and Mark A. Price. *Rete-netzwerk-red: analyzing and visualizing scholarly networks using the Network Workbench Tool*. Scientometrics, vol. 83, no. 3, pages 863–876, 2010. (Cited on page 117.)
- [Bostick 2004] David L Bostick, Min Shen and Iosif I Vaisman. *A simple topological representation of protein structure: implications for new, fast, and robust structural classification*. Proteins, vol. 56, no. 3, pages 487–501, 2004. (Cited on page 29.)
- [Bouker 2012] Slim Bouker, Rabie Saidi, Sadok Ben Yahia and Engelbert Mephu Nguifo. *Ranking and Selecting Association Rules Based on Dominance Relationship*. In Proceedings of the 2012 IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI '12, pages 658–665. IEEE Computer Society, 2012. (Cited on page 54.)
- [Brachman 1996] Ronald J. Brachman and Tej Anand. *The Process of Knowledge Discovery in Databases*. In Advances in Knowledge Discovery and Data Mining, pages 37–57. AAAI/MIT Press, 1996. (Cited on page 11.)
- [Branden 1991] Carl Branden and John Tooze. *Introduction to protein structure*. Garland Publishing, 1991. (Cited on page 22.)
- [Brick 2008] Kevin Brick and Elisabetta Pizzi. *A novel series of compositionally biased substitution matrices for comparing Plasmodium proteins*. BMC Bioinformatics, vol. 9, no. 1, pages 236+, 2008. (Cited on page 26.)
- [Cadenas 2013] José M. Cadenas, M. Carmen Garrido and Raquel MartíNez. *Feature subset selection Filter-Wrapper based on low quality data*. Expert Systems with Applications, vol. 40, no. 16, pages 6241–6252, 2013. (Cited on page 42.)
- [Chen 2008] Chen Chen, Cindy Xide Lin, Xifeng Yan and Jiawei Han. *On effective presentation of graph patterns: a structural representative approach*. In Proceedings of the 17th ACM conference on Information and knowledge management, pages 299–308. ACM, 2008. (Cited on pages 50, 88 and 98.)

- [Cheng 2010] Hong Cheng, Xifeng Yan and Jiawei Han. *Mining Graph Patterns*. In Managing and Mining Graph Data, pages 365–392. Springer, 2010. (Cited on pages 34 and 62.)
- [Cook 1994] Diane J. Cook and Lawrence B. Holder. *Substructure discovery using minimum description length and background knowledge*. Journal of Artificial Intelligence Research, vol. 1, pages 231–255, 1994. (Cited on page 39.)
- [Cook 2006] Diane J. Cook and Lawrence B. Holder. Mining graph data. Wiley, 2006. (Cited on page 34.)
- [Crick 1958] Francis Crick. *On protein synthesis*. In Symposium of the Society of Experimental Biology, volume 12, pages 138–163, 1958. (Cited on page 19.)
- [Crick 1970] Francis Crick. *Central dogma of molecular biology*. Nature, vol. 227, pages 561–563, 1970. (Cited on page 19.)
- [Cuff 2011] Alison L. Cuff, Ian Sillitoe, Tony Lewis, Andrew B. Clegg, Robert Rentzsch, Nicholas Furnham, Marialuisa Pellegrini-Calace, David T. Jones, Janet M. Thornton and Christine A. Orengo. *Extending CATH: increasing coverage of the protein structure universe and linking structure with function*. Nucleic Acids Research, vol. 39, pages 420–426, 2011. (Cited on page 62.)
- [Dayhoff 1978] M. O. Dayhoff, R. M. Schwartz and B. C. Orcutt. *A model of evolutionary change in proteins*. Atlas of protein sequence and structure, vol. 5, no. suppl 3, pages 345–351, 1978. (Cited on page 24.)
- [Delaunay 1934] Boris Delaunay. *Sur la sphère vide. A la mémoire de Georges Vorono*. Bulletin de l’Académie des Sciences de l’URSS. Classe des sciences mathématiques et naturelles, vol. 6, pages 793–800, 1934. (Cited on page 28.)
- [Dempster 1977] A. P. Dempster, N. M. Laird and D. B. Rubin. *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society: Series B, vol. 39, pages 1–38, 1977. (Cited on page 17.)
- [Dhifli 2010] Wajdi Dhifli and Rabie Saidi. *Protein graph repository*. In Extraction et gestion des connaissances (EGC), pages 641–642, 2010. (Cited on page 31.)

- [Dhifli 2012a] Wajdi Dhifli, Rabie Saidi and Engelbert Mephu Nguifo. *Mining Representative Subgraphs by Substitution Matrices: An Application on Protein Structures*. In NIPS workshop on Machine Learning in Computational Biology (MLCB), 2012. (Cited on page 86.)
- [Dhifli 2012b] Wajdi Dhifli, Rabie Saidi and Engelbert Mephu Nguifo. *A novel approach for mining representative spatial motifs of proteins*. In Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine, BCB '12, pages 506–508. ACM, 2012. (Cited on page 86.)
- [Dhifli 2013a] Wajdi Dhifli, Rabie Saidi and Engelbert Mephu Nguifo. *Mining Representative Unsubstituted Graph Patterns Using Prior Similarity Matrix*. In Journée sur la Fouille de Données (GDR I3), 2013. (Cited on page 86.)
- [Dhifli 2013b] Wajdi Dhifli, Rabie Saidi and Engelbert Mephu Nguifo. *Smoothing 3D protein structure motifs through graph mining and amino-acids similarities*. Journal of Computational Biology, page to appear, 2013. (Cited on page 86.)
- [Dhifli 2013c] Wajdi Dhifli, Rabie Saidi and Engelbert Mephu Nguifo Engelbert. *Smoothing 3D protein structure motifs through graph mining and amino-acids similarities*. In Proceedings of the 14<sup>th</sup> Journées Ouvertes en Biologie, Informatique et Mathématiques (JOBIM), pages 65–74, 2013. (Cited on page 86.)
- [Diallo 2009] Abdoulaye Baniré Diallo, Dunarel Badescu, Mathieu Blanchette and Vladimir Makarenkov. *A Whole Genome Study and Identification of Specific Carcinogenic Regions of the Human Papilloma Viruses*. Journal of Computational Biology, vol. 16, no. 10, pages 1461–1473, 2009. (Cited on page 18.)
- [Dobson 2003] Paul D. Dobson and Andrew J. Doig. *Distinguishing Enzyme Structures from Non-enzymes Without Alignments*. Journal of Molecular Biology, vol. 330, no. 4, pages 771–83, 2003. (Cited on page 98.)
- [Doppelt 2007] Olivia Doppelt, Fabrice Moriaud, Aurélie Bornot and Alexandre G. de Brevern. *Functional annotation strategy for protein structures*. Bioinformatics, vol. 9, no. 1, pages 357–359, 2007. (Cited on page 31.)

- [Eddy 2004] Sean R Eddy. *Where did the BLOSUM62 alignment score matrix come from?* Nature Biotechnology, vol. 22, pages 1035–1036, 2004. (Cited on pages 65 and 75.)
- [Ester 1996] Martin Ester, Hans peter Kriegel, Jorg S and Xiaowei Xu. *A density-based algorithm for discovering clusters in large spatial databases with noise*. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pages 226–231. AAAI Press, 1996. (Cited on page 17.)
- [Faust 2010] Karoline Faust, Pierre Dupont, Jérôme Callut and Jacques van Helden. *Pathway discovery in metabolic networks by subgraph extraction*. Bioinformatics, vol. 26, no. 9, pages 1211–1218, 2010. (Cited on page 63.)
- [Fayyad 1996] Usama M. Fayyad, Gregory Piatetsky-Shapiro and Padhraic Smyth. Advances in knowledge discovery and data mining, chapter From data mining to knowledge discovery: an overview, pages 1–34. American Association for Artificial Intelligence, 1996. (Cited on page 10.)
- [Fayyad 1997] Usama M. Fayyad. *Knowledge Discovery in Databases: An Overview*. In Inductive Logic Programming ILP, pages 3–16, 1997. (Cited on pages 10 and 11.)
- [Fei 2010] Hongliang Fei and Jun Huan. *Boosting with structure information in the functional space: an application to graph classification*. In ACM knowledge discovery and data mining conference (KDD), pages 643–652, 2010. (Cited on pages 53, 73, 81 and 97.)
- [Gamblin 2010] Todd Gamblin, Bronis R. de Supinski, Martin Schulz, Rob Fowler and Daniel A. Reed. *Clustering performance data efficiently at massive scales*. In Proceedings of the 24th ACM International Conference on Supercomputing, ICS '10, pages 243–252. ACM, 2010. (Cited on page 114.)
- [Geng 2006] Liqiang Geng and Howard J. Hamilton. *Interestingness measures for data mining: A survey*. ACM Computing Surveys, vol. 38, no. 3, 2006. (Cited on page 14.)
- [Gibert 2012] Jaume Gibert, Ernest Valveny and Horst Bunke. *Feature selection on node statistics based embedding of graphs*. Pattern Recognition Letters, vol. 33, no. 15, pages 1980 – 1990, 2012. (Cited on page 89.)

- [Grünwald 2007] Peter D. Grünwald. The minimum description length principle. Adaptive computation and machine learning. The MIT Press, 2007. (Cited on page 71.)
- [Guyon 2003] Isabelle Guyon and Andre Elisseeff. *An Introduction to Variable and Feature Selection*. Journal of Machine Learning Research, vol. 3, pages 1157–1182, 2003. (Cited on page 42.)
- [Han 2006] Jiawei Han and Micheline Kamber. Data mining: Concepts and techniques (2nd edition). Morgan Kaufmann, 2 edition, 2006. (Cited on pages 10 and 11.)
- [Hasan 2007] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, Jeremy Besson and Mohammed J. Zaki. *ORIGAMI: Mining Representative Orthogonal Graph Patterns*. In Proceedings of the 2007 Seventh IEEE International Conference on Data Mining, pages 153–162. IEEE Computer Society, 2007. (Cited on pages 48, 64 and 88.)
- [Hasan 2009] Mohammad Al Hasan and Mohammed J. Zaki. *Output Space Sampling for Graph Patterns*. Proceedings of the VLDB Endowment (35th International Conference on Very Large Data Bases), vol. 2, no. 1, pages 730–741, 2009. (Cited on pages 48 and 62.)
- [Henikoff 1992] Steven Henikoff and Jorja G. Henikoff. *Amino acid substitution matrices from protein blocks*. Proceedings of The National Academy of Sciences, vol. 89, pages 10915–10919, 1992. (Cited on page 24.)
- [Hochbaum 1997] Dorit S. Hochbaum, editor. Approximation algorithms for NP-hard problems. PWS Publishing Co., 1997. (Cited on page 50.)
- [Holm 2010] Liisa Holm and Päivi Rosenström. *Dali server: conservation mapping in 3D*. Nucleic Acids Research, vol. 38, no. Web-Server-Issue, pages 545–549, 2010. (Cited on page 83.)
- [Huan 2003] Jun Huan, Wei Wang and Jan Prins. *Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism*. In IEEE International Conference on Data Mining (ICDM), pages 549–552, 2003. (Cited on pages 40 and 63.)
- [Huan 2004a] Jun Huan, Wei Wang, Deepak B, Jack Snoeyink, Jan Prins and Alex Tropsha. *Mining spatial motifs from protein structure graphs*. In International Conference on Research in Computational Molecular Biology (RECOMB), pages 308–315, 2004. (Cited on page 62.)

- [Huan 2004b] Jun Huan, Wei Wang, Jan Prins and Jiong Yang. *SPIN: mining maximal frequent subgraphs from graph databases*. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 581–586. ACM, 2004. (Cited on page 41.)
- [Huan 2005] Jun Huan, Deepak Bandyopadhyay, Wei Wang, Jack Snoeyink, Jan Prins and Alexander Tropsha. *Comparing Graph Representations of Protein Structure for Mining Family-Specific Residue-Based Packing Motifs*. Journal of Computational Biology, vol. 12, no. 6, pages 657–671, 2005. (Cited on pages 29, 30, 75 and 99.)
- [Ingram 2006] Piers J. Ingram, Michael P. H. Stumpf and Jaroslav Stark. *Network motifs: structure does not determine function*. BMC Genomics, vol. 7, no. 1, pages 1–12, 2006. (Cited on page 91.)
- [Inokuchi 2000] Akihiro Inokuchi, Takashi Washio and Hiroshi Motoda. *An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data*. In The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pages 13–23, 2000. (Cited on page 39.)
- [Jain 2010] Anil K. Jain. *Data clustering: 50 years beyond K-means*. Pattern Recognition Letters, vol. 31, no. 8, pages 651 – 666, 2010. (Cited on pages 17 and 94.)
- [Jiang 2013] Chuntao Jiang, Frans Coenen and Michele Zito. *A survey of frequent subgraph mining algorithms*. The Knowledge Engineering Review, vol. 28, pages 75–105, 2013. (Cited on page 39.)
- [Jin 2009] Ning Jin, Calvin Young and Wei Wang. *Graph classification based on pattern co-occurrence*. In ACM International Conference on Information and Knowledge Management, pages 573–582, 2009. (Cited on pages 52, 62 and 81.)
- [Jin 2010] Ning Jin, Calvin Young and Wei Wang. *GAIA: graph classification using evolutionary computation*. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, SIGMOD ’10, pages 879–890. ACM, 2010. (Cited on page 51.)
- [Kaufman 1987] Leonard Kaufman and Peter Rousseeuw. *Clustering by means of medoids*. Reports of the Faculty of Mathematics and Informatics. Delft University of Technology. Fac., Univ., 1987. (Cited on pages 17, 94 and 95.)

- [Kaufman 1990] Leonard Kaufman and Peter J. Rousseeuw. Finding groups in data: An introduction to cluster analysis. Wiley-Interscience, 9th edition, 1990. (Cited on page 95.)
- [King 2001] Ross D. King, Ashwin Srinivasan and Luc Dehaspe. *Warmr: a data mining tool for chemical data*. Journal of Computer-Aided Molecular Design, vol. 15, no. 2, pages 173–181, 2001. (Cited on page 39.)
- [Kleywegt 1999] Gerard J. Kleywegt. *Recognition of spatial motifs in protein structures*. Journal of molecular biology, vol. 285, no. 4, pages 1887–1897, 1999. (Cited on pages 31 and 62.)
- [Knabe 2008] Johannes F. Knabe, Chrystopher L. Nehaniv and Maria J. Schilstra. *Do motifs reflect evolved function?-No convergent evolution of genetic regulatory network subgraph topologies*. Biosystems, vol. 94, no. 1-2, pages 68 – 74, 2008. (Cited on page 91.)
- [Krishna 2011] Varun Krishna, N.N.R. Ranga Suri and G Athithan. *A comparative survey of algorithms for frequent subgraph discovery*. Current Science, vol. 100, no. 2, pages 190–198, 2011. (Cited on page 39.)
- [Kuramochi 2001] Michihiro Kuramochi and George Karypis. *Frequent Subgraph Discovery*. In IEEE International Conference on Data Mining (ICDM), pages 313–320, 2001. (Cited on page 39.)
- [Ladha 2011] L. Ladha and T. Deepa. *Feature selection methods and algorithms*. International Journal on Computer Science and Engineering (IJCSE), vol. 5, no. 5, pages 1787–1797, 2011. (Cited on pages 42 and 43.)
- [Lakshmi 2012] K. Lakshmi and T Meyyappan. *Frequent Subgraph Mining Algorithms - A Survey and Framework for Classification*. International Journal of Information Technology Convergence and Services (IJITCS), vol. 2, pages 189–202, 2012. (Cited on page 39.)
- [Lazar 2012] Cosmin Lazar, Jonatan Taminiau, Stijn Meganck, David Steenhoff, Alain Coletta, Colin Molter, Virginie de Schaetzen, Robin Duque, Hugues Bersini and Ann Nowe. *A Survey on Filter Techniques for Feature Selection in Gene Expression Microarray Analysis*. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), vol. 9, no. 4, pages 1106–1119, 2012. (Cited on page 43.)
- [Leskovec 2005] Jure Leskovec, Jon Kleinberg and Christos Faloutsos. *Graphs over time: densification laws, shrinking diameters and possible explanations*. In Proceedings of the eleventh ACM SIGKDD international

- conference on Knowledge discovery in data mining, KDD '05, pages 177–187. ACM, 2005. (Cited on pages 89 and 91.)
- [Li 2007] Jinyan Li, Guimei Liu, Haiquan Li and Limsoon Wong. *Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms*. IEEE Trans. on Knowl. and Data Eng., vol. 19, no. 12, pages 1625–1637, 2007. (Cited on page 88.)
- [Li 2008] Peipei Li, Xuegang Hu and Xindong Wu. *Mining Concept-Drifting Data Streams with Multiple Semi-Random Decision Trees*. In Proceedings of the 4th international conference on Advanced Data Mining and Applications, ADMA '08, pages 733–740. Springer-Verlag, 2008. (Cited on page 16.)
- [Li 2010] Yuhua Li, Quan Lin, Ruixuan Li and Dongsheng Duan. *TGP: mining top-K frequent closed graph pattern without minimum support*. In Proceedings of the 6th international conference on Advanced data mining and applications (ADMA), pages 537–548. Springer-Verlag, 2010. (Cited on page 46.)
- [Li 2012] Geng Li, Murat Semerci, Bulent Yener and Mohammed J. Zaki. *Effective graph classification based on topological and label attributes*. Statistical Analysis and Data Mining, vol. 5, no. 4, pages 265–283, 2012. (Cited on pages 89, 91 and 93.)
- [Liu 1998] Huan Liu and Hiroshi Motoda. Feature extraction, construction and selection: A data mining perspective. Kluwer Academic Publishers, Norwell, MA, USA, 1998. (Cited on page 42.)
- [Liu 2007a] Huan Liu and Hiroshi Motoda. Computational methods of feature selection. Data Mining and Knowledge Discovery. Chapman and Hall/CRC, 2007. (Cited on page 42.)
- [Liu 2007b] Yong Liu, Jianzhong Li, Jinghua Zhu and Hong Gao. *Clustering Frequent Graph Patterns*. Journal of Digital Information Management, vol. 5, no. 6, pages 335–346, 2007. (Cited on page 46.)
- [Lovell 2003] Simon C. Lovell, Ian W. Davis, W. Bryan Arendall, Paul I. W. de Bakker, J. Michael Word, Michael G. Prisant, Jane S. Richardson and David C. Richardson. *Structure validation by  $\alpha$  geometry:  $\Phi$ ,  $\Psi$  and  $\omega$  deviation*. Proteins: Structure, Function, and Bioinformatics, vol. 50, no. 3, pages 437–450, 2003. (Cited on page 29.)



- [MacQueen 1967] J. B. MacQueen. *Some Methods for Classification and Analysis of MultiVariate Observations*. In L. M. Le Cam and J. Neyman, editors, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281–297. University of California Press, 1967. (Cited on pages 17 and 96.)
- [Mannila 1997] Heikki Mannila. *Methods and Problems in Data Mining*. In Proceedings of the 6th International Conference on Database Theory, ICDT '97, pages 41–55. Springer-Verlag, 1997. (Cited on page 11.)
- [Mavridis 2010] Lazaros Mavridis and David W. Ritchie. *3D-Blast: 3D Protein Structure Alignment, Comparison, and Classification Using Spherical Polar Fourier Correlations*. In Pacific Symposium on Bio-computing, pages 281–292, 2010. (Cited on page 62.)
- [McGarry 2005] Ken McGarry. *A survey of interestingness measures for knowledge discovery*. The Knowledge Engineering Review, vol. 20, no. 1, pages 39–61, 2005. (Cited on page 14.)
- [Morgat 2012] Anne Morgat, Eric Coissac, Elisabeth Coudert, Kristian B. Axelsen, Guillaume Keller, Amos Bairoch, Alan Bridge, Lydie Bouguet-eret, Ioannis Xenarios and Alain Viari. *UniPathway: a resource for the exploration and annotation of metabolic pathways*. Nucleic Acids Research, vol. 40, no. Database-Issue, pages 761–769, 2012. (Cited on page 18.)
- [Mount 2008] David W. Mount. *Comparison of the PAM and BLOSUM Amino Acid Substitution Matrices*. Cold Spring Harbor Protocols, no. 7, pages 59+, 2008. (Cited on page 26.)
- [Ng 1994] Raymond T. Ng and Jiawei Han. *Efficient and Effective Clustering Methods for Spatial Data Mining*. In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, pages 144–155. Morgan Kaufmann Publishers Inc., 1994. (Cited on page 95.)
- [Ng 2002] Raymond T. Ng and Jiawei Han. *CLARANS: A Method for Clustering Objects for Spatial Data Mining*. IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 5, pages 1003–1016, 2002. (Cited on pages 50 and 95.)
- [Nijssen 2001] Siegfried Nijssen and Joost Kok. *Faster association rules for multiple relations*. In Proceedings of the 17th international joint conference on Artificial intelligence, IJCAI'01, pages 891–896. Morgan Kaufmann Publishers Inc., 2001. (Cited on page 39.)

- [Nijssen 2004] Siegfried Nijssen and Joost N. Kok. *A quickstart in frequent structure mining can make a difference*. In ACM knowledge discovery and data mining conference (KDD), pages 647–652, 2004. (Cited on pages 40 and 63.)
- [Papadopoulos 2008] Apostolos N. Papadopoulos, Apostolos Lyritsis and Yannis Manolopoulos. *SkyGraph: an algorithm for important subgraph discovery in relational graphs*. Data Mining and Knowledge Discovery, vol. 17, no. 1, pages 57–76, 2008. (Cited on page 54.)
- [Pennerath 2009] Frédéric Pennerath and Amedeo Napoli. *The Model of Most Informative Patterns and Its Application to Knowledge Extraction from Graph Databases*. In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II, ECML PKDD '09, pages 205–220. Springer-Verlag, 2009. (Cited on page 54.)
- [Quinlan 1986] John Ross Quinlan. *Induction of Decision Trees*. Machine Learning, vol. 1, no. 1, pages 81–106, 1986. (Cited on page 16.)
- [Quinlan 1993] John Ross Quinlan. C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., 1993. (Cited on page 16.)
- [Ranu 2009] Sayan Ranu and Ambuj K. Singh. *GraphSig: A Scalable Approach to Mining Significant Subgraphs in Large Graph Databases*. In IEEE 25th International Conference on Data Engineering., pages 844–855, 2009. (Cited on page 51.)
- [Ranu 2012] Sayan Ranu and Ambuj K. Singh. *Indexing and mining topological patterns for drug discovery*. In Proceedings of the 15th International Conference on Extending Database Technology, EDBT '12, pages 562–565. ACM, 2012. (Cited on page 89.)
- [Recupero 2008] Diego Reforgiato Recupero, Rodrigo A. Gutiérrez and Dennis Shasha. *Graphclust: a Method for Clustering Database of Graphs*. Journal of Information & Knowledge Management, vol. 7, no. 4, pages 231–241, 2008. (Cited on page 117.)
- [Rissanen 1978] Jorma Rissanen. *Modeling by shortest data description*. Automata, vol. 14, no. 5, pages 465 – 471, 1978. (Cited on page 71.)
- [Ritchie 2010] David W. Ritchie and Vishwesh Venkatraman. *Ultra-fast FFT protein docking on graphics processors*. Bioinformatics, vol. 26, no. 19, pages 2398–2405, 2010. (Cited on page 18.)

- [Rodenacker 1990] Karsten Rodenacker and Paul Bischoff. *Quantification of tissue sections: Graph theory and topology as modelling tools*. Pattern Recognition Letters, vol. 11, no. 4, pages 275 – 284, 1990. (Cited on page 89.)
- [Saeys 2007] Yvan Saeys, Iñaki Inza and Pedro Larrañaga. *A review of feature selection techniques in bioinformatics*. Bioinformatics, vol. 23, no. 19, pages 2507–2517, 2007. (Cited on page 42.)
- [Saidi 2009] Rabie Saidi, Mondher Maddouri and Engelbert Mephu Nguifo. *Comparing graph-based representations of protein for mining purposes*. In Proceedings of the ACM KDD Workshop on Statistical and Relational Learning in Bioinformatics, pages 35–38, 2009. (Cited on pages 28 and 29.)
- [Saidi 2010] Rabie Saidi, Mondher Maddouri and Engelbert Mephu Nguifo. *Protein sequences classification by means of feature extraction with substitution matrices*. BMC Bioinformatics, vol. 11, no. 1, pages 175+, 2010. (Cited on page 66.)
- [Saidi 2012] Rabie Saidi, Wajdi Dhifli, Mondher Maddouri and Engelbert Mephu Nguifo. *A novel approach of spatial motif extraction to classify protein structures*. In Proceedings of the 13<sup>th</sup> Journées Ouvertes en Biologie, Informatique et Mathématiques (JOBIM), pages 209–216, 2012. (Cited on page 55.)
- [Saigo 2008] Hiroto Saigo, Nicole Krämer and Koji Tsuda. *Partial least squares regression for graph mining*. In ACM knowledge discovery and data mining conference (KDD), pages 578–586, 2008. (Cited on pages 52 and 81.)
- [Saigo 2009] Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo and Koji Tsuda. *gBoost: a mathematical programming approach to graph classification and regression*. Machine Learning, vol. 75, no. 1, pages 69–89, 2009. (Cited on page 53.)
- [Schietgat 2011] Leander Schietgat, Fabrizio Costa, Jan Ramon and Luc Raedt. *Effective feature construction by maximum common subgraph sampling*. Machine Learning, vol. 83, no. 2, pages 137–161, 2011. (Cited on page 49.)
- [Sheikh 2007] Yaser Sheikh, Erum A. Khan and Takeo Kanade. *Mode-seeking by Medoidshifts*. In Proceedings of the 11th IEEE International Con-

- ference on Computer Vision, ICCV '07, pages 1–8. IEEE, 2007. (Cited on page 114.)
- [Simoudis 1996] Evangelos Simoudis. *Reality Check for Data Mining*. IEEE Expert, vol. 11, no. 5, pages 26–33, 1996. (Cited on page 11.)
- [Soulet 2011] Arnaud Soulet, Chedy Raïssi, Marc Plantevit and Bruno Cremilleux. *Mining Dominant Patterns in the Sky*. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining, ICDM '11, pages 655–664. IEEE Computer Society, 2011. (Cited on page 54.)
- [Stigler 1986] Stephen M. Stigler. The history of statistics: the measurement of uncertainty before 1900. Belknap Press of Harvard University Press, 1986. (Cited on page 16.)
- [Stout 2008] Michael Stout, Jaume Bacardit, Jonathan D. Hirst, Robert E. Smith and Natalio Krasnogor. *Prediction of topological contacts in proteins using learning classifier systems*. Soft Computing, vol. 13, pages 245–258, 2008. (Cited on page 29.)
- [Sun 2012] Hong Sun, Ahmet Sacan, Hakan Ferhatosmanoglu and Yusu Wang. *Smolign: A Spatial Motifs-Based Protein Multiple Structural Alignment Method*. IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 9, no. 1, pages 249–261, 2012. (Cited on page 62.)
- [Takigawa 2011] Ichigaku Takigawa and Hiroshi Mamitsuka. *Efficiently mining  $\delta$ -tolerance closed frequent subgraphs*. Mach. Learn., vol. 82, no. 2, pages 95–121, 2011. (Cited on page 88.)
- [Theocharidis 2009] Athanasios Theocharidis, Stjin van Dongen, Anton J. Enright and Tom C. Freeman. *Network visualization and analysis of gene expression data using BioLayout Express(3D)*. Nature protocols, vol. 4, no. 10, pages 1535–1550, 2009. (Cited on page 117.)
- [Thoma 2010] Marisa Thoma, Hong Cheng, Arthur Gretton, Jiawei Han, Hans-Peter Kriegel, Alex Smola, Le Song, Philip S. Yu, Xifeng Yan and Karsten M. Borgwardt. *Discriminative frequent subgraph mining with optimality guarantees*. Statistical Analysis and Data Mining, vol. 3, no. 5, pages 302–318, 2010. (Cited on pages 51, 63, 64 and 98.)
- [Thomas 2006] Lini T. Thomas, Satyanarayana R. Valluri and Kamalakara Karlapalem. *MARGIN: Maximal Frequent Subgraph Mining*. In Sixth International Conference on Data Mining (ICDM)., pages 1097–1101, 2006. (Cited on pages 41, 64 and 88.)

- [Tong 2012] Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos and Christos Faloutsos. *Gelling, and melting, large graphs by edge manipulation*. In Proceedings of the 21st ACM international conference on Information and knowledge management, CIKM '12, pages 245–254. ACM, 2012. (Cited on page 89.)
- [Vallabhajosyula 2009] Ravishankar R. Vallabhajosyula, Deboki Chakravarti, Samina Lutfeali, Animesh Ray and Alpan Raval. *Identifying Hubs in Protein Interaction Networks*. PLoS ONE, vol. 4, no. 4, page e5344, 2009. (Cited on page 63.)
- [Vanetik 2002] Natalia Vanetik, Ehud Gudes and Solomon Eyal Shimony. *Computing Frequent Graph Patterns from Semistructured Data*. In Proceedings of the IEEE International Conference on Data Mining, pages 458–465. IEEE Computer Society, 2002. (Cited on page 39.)
- [Vapnik 1995] Vladimir Vapnik and Corinna Cortes. *Support-Vector Networks*. Machine Learning, vol. 20, pages 273–297, 1995. (Cited on page 16.)
- [Veeramalai 2008] Mallika Veeramalai and David Gilbert. *A novel method for comparing topological models of protein structures enhanced with ligand information*. Bioinformatics, vol. 24, no. 23, pages 2698–2705, 2008. (Cited on page 89.)
- [Viari 2003] Alain Viari. *How Does Computer Science Change Molecular Biology?* In Helmut Alt and Michel Habib, editors, STACS, volume 2607 of *Lecture Notes in Computer Science*, page 13. Springer Berlin Heidelberg, 2003. (Cited on page 18.)
- [Vishveshwara 2002] Saraswathi Vishveshwara, K. V. Brinda and N. Kannan. *Protein structure: insights from graph theory*. Journal of Theoretical and Computational Chemistry, vol. 1, no. 1, pages 187–212, 2002. (Cited on page 62.)
- [von Öhsen 2004] Niklas von Öhsen, Ingolf Sommer, Ralf Zimmer and Thomas Lengauer. *Arby: automatic protein structure prediction using profile-profile alignment and confidence measures*. Bioinformatics, vol. 20, no. 14, pages 2228–2235, 2004. (Cited on page 62.)
- [Wasserman 2010] L. Wasserman. *All of statistics: A concise course in statistical inference*. Springer Texts in Statistics. Springer, 2010. (Cited on page 16.)

- [Witten 2005] Ian H Witten and Eibe Frank. Data mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005. (Cited on page 75.)
- [Wolpert 1995] David Wolpert and William Macready. *No Free Lunch Theorems for Search*. Rapport technique, Santa Fe Institute, 1995. (Cited on page 16.)
- [Xin 2006] Dong Xin, Hong Cheng, Xifeng Yan and Jiawei Han. *Extracting redundancy-aware top-k patterns*. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 444–453. ACM, 2006. (Cited on page 45.)
- [Yan 2002] Xifeng Yan and Jiawei Han. *gSpan: Graph-based substructure pattern mining*. In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM), volume 02, pages 721–724. IEEE Computer Society, 2002. (Cited on pages 40, 63, 75 and 99.)
- [Yan 2003] Xifeng Yan and Jiawei Han. *CloseGraph: mining closed frequent graph patterns*. In ACM knowledge discovery and data mining conference (KDD), pages 286–295, 2003. (Cited on pages 41, 64 and 88.)
- [Yan 2008] Xifeng Yan, Hong Cheng, Jiawei Han and Philip S. Yu. *Mining significant graph patterns by leap search*. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD '08, pages 433–444. ACM, 2008. (Cited on pages 51, 73, 81 and 97.)
- [Yu 2005] Yi-Kuo Yu and Stephen F. Altschul. *The construction of amino acid substitution matrices for the comparison of proteins with non-standard compositions*. Bioinformatics, vol. 21, no. 7, pages 902–911, 2005. (Cited on page 26.)
- [Zaki 2014] Mohammed J. Zaki and Jr. Wagner Meira. Data mining and analysis: Foundations and algorithms, chapter Graph Pattern Mining, pages 285–302. Cambridge University Press, 2014. (Cited on pages 10 and 35.)
- [Zhang 1996] Tian Zhang, Raghu Ramakrishnan and Miron Livny. *BIRCH: an efficient data clustering method for very large databases*. In Proceedings of the 1996 ACM SIGMOD international conference on Management of data, SIGMOD '96, pages 103–114. ACM, 1996. (Cited on page 17.)

- [Zhang 2009] Shijie Zhang, Jiong Yang and Shirong Li. *RING: An Integrated Method for Frequent Representative Subgraph Mining*. In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, pages 1082–1087. IEEE Computer Society, 2009. (Cited on page 47.)
- [Zhu 2012] Yuanyuan Zhu, Jeffrey Xu Yu, Hong Cheng and Lu Qin. *Graph classification: a diversified discriminative feature selection approach*. In Proceedings of the 21st ACM international conference on Information and knowledge management, CIKM '12, pages 205–214. ACM, 2012. (Cited on page 52.)
- [Zimmermann 2010] Karel Zimmermann and Jean-François Gibrat. *Amino acid "little Big Bang": Representing amino acid substitution matrices as dot products of Euclidian vectors*. BMC Bioinformatics, vol. 11, no. 1, pages 4+, 2010. (Cited on page 26.)

---

## Topological and Domain Knowledge-based Subgraph Mining: Application on Protein 3D-Structures

**Abstract:** This thesis is in the intersection of two proliferating research fields, namely data mining and bioinformatics. With the emergence of graph data in the last few years, many efforts have been devoted to mining frequent subgraphs from graph databases. Yet, the number of discovered frequent subgraphs is usually exponential, mainly because of the combinatorial nature of graphs. Many frequent subgraphs are irrelevant because they are redundant or just useless for the user. Besides, their high number may hinder and even makes further explorations unfeasible. Redundancy in frequent subgraphs is mainly caused by structural and/or semantic similarities, since most discovered subgraphs differ slightly in structure and may infer similar or even identical meanings.

In this thesis, we propose two approaches for selecting representative subgraphs among frequent ones in order to remove redundancy. Each of the proposed approaches addresses a specific type of redundancy. The first approach focuses on semantic redundancy where similarity between subgraphs is measured based on the similarity between their nodes' labels, using prior domain knowledge. The second approach focuses on structural redundancy where subgraphs are represented by a set of user-defined topological descriptors, and similarity between subgraphs is measured based on the distance between their corresponding topological descriptions.

The main application data of this thesis are protein 3D-structures. This choice is based on biological and computational reasons. From a biological perspective, proteins play crucial roles in almost every biological process. They are responsible of a variety of physiological functions. From a computational perspective, we are interested in mining complex data. Proteins are a perfect example of such data as they are made of complex structures composed of interconnected amino acids which themselves are composed of interconnected atoms. Large amounts of protein structures are currently available in online databases, in computer analyzable formats. Protein 3D-structures can be transformed into graphs where amino acids are the graph nodes and their connections are the graph edges. This enables using graph mining techniques to study them. The biological importance of proteins, their complexity, and their availability in computer analyzable formats made them a perfect application data for this thesis.

Implementation of the research works are available on my personal home page <http://fc.isima.fr/~dhiffi> or upon email request.

**Keywords:** Feature selection, pattern mining, frequent subgraph, representative unsubstituted subgraph, topological representative subgraph, protein structure

---



---

## Fouille de Sous-graphes Basée sur la Topologie et la Connaissance du Domaine: Application sur les Structures 3D de Protéines

**Résumé:** Cette thèse est à l'intersection de deux domaines de recherche en plein expansion, à savoir la fouille de données et la bioinformatique. Avec l'émergence des bases de graphes au cours des dernières années, de nombreux efforts ont été consacrés à la fouille des sous-graphes fréquents. Mais le nombre de sous-graphes fréquents découverts est exponentiel, cela est due principalement à la nature combinatoire des graphes. Beaucoup de sous-graphes fréquents ne sont pas pertinents parce qu'ils sont redondants ou tout simplement inutiles pour l'utilisateur. En outre, leur nombre élevé peut nuire ou même rendre parfois irréalisable toute utilisation ultérieure. La redondance dans les sous-graphes fréquents est principalement due à la similarité structurelle et / ou sémantique, puisque la plupart des sous-graphes découverts diffèrent légèrement dans leur structures et peuvent exprimer des significations similaires ou même identiques.

Dans cette thèse, nous proposons deux approches de sélection des sous-graphes représentatifs parmi les fréquents afin d'éliminer la redondance. Chacune des approches proposées s'intéresse à un type spécifique de redondance. La première approche s'adresse à la redondance sémantique où la similarité entre les sous-graphes est mesurée en fonction de la similarité entre les étiquettes de leurs noeuds, en utilisant les connaissances de domaine. La deuxième approche s'adresse à la redondance structurelle où les sous-graphes sont représentés par des descripteurs topologiques définis par l'utilisateur, et la similarité entre les sous-graphes est mesurée en fonction de la distance entre leurs descriptions topologiques respectives.

Les principales données d'application de cette thèse sont les structures 3D des protéines. Ce choix repose sur des raisons biologiques et informatiques. D'un point de vue biologique, les protéines jouent un rôle crucial dans presque tous les processus biologiques. Ils sont responsables d'une variété de fonctions physiologiques. D'un point de vue informatique, nous sommes intéressés à la fouille de données complexes. Les protéines sont un exemple parfait de ces données car elles sont faites de structures complexes composées d'acides aminés interconnectés qui sont eux-mêmes composées d'atomes interconnectés. Des grandes quantités de structures protéiques sont actuellement disponibles dans les bases de données en ligne. Les structures 3D des protéines peuvent être transformées en graphes où les acides aminés représentent les noeuds du graphe et leurs connexions représentent les arêtes. Cela permet d'utiliser des techniques de fouille de graphes pour les étudier. L'importance biologique des protéines et leur complexité ont fait d'elles des données d'application appropriées pour cette thèse.

Les implémentations des ces travaux de recherche sont disponibles sur ma page personnelle <http://fc.isima.fr/~dhifli> ou sur demande par courriel.

**Mots clés:** Sélection de motifs, fouille de motifs, sous-graphe fréquent, sous-graphe représentant non-substitué, graphe représentant topologique, structure de protéine

---