Université de Tunis

Institut Supérieur de Gestion

MASTER RECHERCHE

Spécialité

Sciences et Techniques de l'Informatique Décisionnelle (STID)

Option

Informatique et Gestion de la Connaissance

Community Detection for Graph-Based Similarity of Protein Binding Pockets

---

Sabrine Mallek

Sous la direction de:

Pr. Zied Elouedi (ISG Tunis, Tunisie)
Dr. Imen Boukhris (ENSI, La manouba , Tunisie)

Laboratoire: LARODEC

Acknowledgements

I offer my deep gratitude to my master thesis advisor Professor Zied Elouedi for his willingness to supervise this work through his suggestions for improvement and his continuous encouragement and useful comments. One simply could not wish for a better supervisor.

I am also very grateful to my co-advisor Doctor Imen Boukhris for her continuous guidance and for spending time reading this thesis and providing useful suggestions and criticism.

May this modest work satisfy you and express my gratitude to the help and advice you have given me.

Special thanks are given to all the professors of Institut Supérieur de Gestion de Tunis and to the members of my laboratory LARODEC.

Contents

General introduction **1**

1 Protein function prediction **3**

1.1 Introduction 3

1.2 Basic concepts 4

1.2.1 Bioinformatics 4

1.2.2 Proteome and proteomics 4

1.2.3 Proteins 5

1.2.4 Protein structure 6

1.2.5 Structure data sources 8

1.3 Major approaches for predicting protein functions 9

1.3.1 Sequence-based methods 9

1.3.2 Structure-based methods 10

1.3.3 Protein interactions-based methods 12

1.4 Protein binding sites 13

1.4.1 Binding sites presentation 13

1.4.2 Binding sites detection 14

1.4.3 Protein function prediction using binding sites 14
1.4.4 Binding sites representation 15
1.5 Conclusion 16

2 Graph similarity 17
2.1 Introduction 17
2.2 Graph matching 18
2.2.1 Exact matching 18
2.2.2 Inexact matching using graph edit distance 23
2.3 Protein function prediction using graph representation of binding sites . . . 25
2.3.1 Graph representation of binding sites 25
2.3.2 Prediction using graph matching 26
2.3.3 Quasi-clique 27
2.4 Conclusion 32

3 Graph-based similarity derived from community detection 33
3.1 Introduction 33
3.2 Definitions and notations 34
3.2.1 From cliques, quasi-cliques to communities 34
3.2.2 Basic concepts 35
3.3 The community detection based approach 36
3.3.1 Graph-based similarity using community detection 36
3.3.2 Similarity measure based on the densest maximum common community 38
3.3.3 Complexity 39
3.3.4 Our contributions 39
3.4 Illustration example 40

3.4.1 The product graph construction step 40
3.4.2 The communities detection step 40
3.4.3 The cliques elimination step 45
3.4.4 The maximum densest common community $MDCC$ extraction step 46
3.4.5 The similarity computation step 46
3.5 Conclusion 47

4 Experimental study 50
4.1 Introduction 50
4.2 Implementation 50
4.2.1 Framework 51
4.2.2 Principle variables 51
4.2.3 Main programs 53
4.2.4 Graph-based similarity derived from the communities detection algorithms 54
4.3 Experimentations 59
4.3.1 Evaluation criteria 59
4.3.2 Experimentation on real datasets 60
4.4 Conclusion 63

General conclusion and future works 65

A NADH-ATP dataset 68
A.1 Introduction 68
A.2 Nicotinamide adenine dinucleotide -NADH 68
A.3 Adenosine triphosphate -ATP 69
A.4 Conclusion 70

References **71**

List of Figures

1.1 Amino acid parts 5

1.2 Four protein structures 7

1.3 Protein binding sites representation within CavBase 12

1.4 3D visualization of the endonuclease (an enzyme) and its binding sites (highlighted in blue) 14

2.1 A possible *mcs* (c) of two graphs (a) and (b) 21

2.2 A minimum common supergraph (d) for a set of three graphs (a), (b) and (c) 21

2.3 A product graph (c) of two graphs (a) and (b) 23

2.4 An edit path (c) for two graphs (a) and (b) 24

2.5 Graph representing protein binding sites 26

2.6 Prediction based on binding sites using graph matching 27

2.7 0.5-quasi-clique 28

2.8 Maximum γ-quasi-clique detection 31

3.1 Illustration of the community detection algorithm operating on a protein binding site product graph 41

3.2 Maximum clique (a) and quasi-clique (b) detection 47

4.1 Accuracy of the four methods 62

A.1 Nicotinamide adenine dinucleotide -NADH 69

A.2 Adenosine triphosphate- ATP 69

List of Tables

1.1 Proteins primary functions 6

4.1 Description of the database 60

4.2 Accuracy of the four methods 61

4.3 Time complexity of the four methods 63

General introduction

Proteins are very essential items of cell life as they regulate all biological processes in the organism. An accurate prediction of the protein functional activity can realize an extensive investment in computational biology and promising opportunities to better understand the cellular mechanism. One can elucidate the protein function by analyzing the structure and physicochemical composition of its binding pocket. Indeed, the latter is physically and chemically complementary to the binding partners and hence its analysis can provide perception to the function of the respective protein.

One can represent the geometry and the physicochemical properties of these protein binding pockets by use of node-labeled and edge-weighted graphs. In fact, graphs are versatile tools that allow the conservation of the structure of objects that are being modeled without loss of information.

One way to predict a protein unknown function is to compare its binding cavities structures with other known collected proteins in one belonging database (e.g. PDB) and detect common regions. This task can be carried out using graph matching, a problem known from graph theory. Graph matching techniques allow the comparison between two graphs according to their common substructure (the maximum common subgraph isomorphism (*mcs*)).

The *mcs* identification can be reformulated into another graph theory problem: the detection of the maximum clique on the product graph. The latter is build upon the two graphs being considered and allows the conservation of the nodes and edges structures. Yet, the clique concept is too restrictive especially in the context of real world molecular data where noise and distortions occur frequently due to mutations and the flexible nature of the protein 3D-structure. Another way to handle the problem of the *mcs* detection consists in replacing it by the quasi-clique concept. Even though the quasi-clique allows more tolerance, its mining is more complicated and is time consuming thus working on

large graphs is a difficult task.

In this master thesis, we propose to optimize the size of the common substructure and make its detection more practical on large graphs. We use an alternative concept, instead of the maximum clique and the maximum quasi-clique, which we call the maximum densest common subgraph. Our approach is based on the notion of community detection on graphs. The maximum almost common subgraph measure is used to assess the similarity between proteins binding sites and therefore predict the functional class of a new protein. Note that our approach is based only on structural information and is independent of sequence or fold homology.

This report is organized as follows:

- Chapter 1 presents the necessary theoretical aspects concerning the basics of protein function prediction.
- Chapter 2 details the graph-based similarity concept and the types of graph matching.
- Chapter 3 describes our new approach based on the community concept and shows how to use it as a similarity measure between two protein biding sites graphs.
- Chapter 4 is related to the implementation phase in which tests on real world data have been performed in order to analyze and evaluate results given by our proposed method.
- Finally, a general conclusion gives a summary of the results achieved in this thesis and presents possible future developments.
- An appendix is provided at the end of this report which presents a description of the dataset used in the experimental phase.

Chapter 1

Protein function prediction

1.1 Introduction

The tissue of the organism is made up by cells and products coded by genes. These products are called proteins, they form the proteome of the genome of the organism and carry out several vital functions which may be either structural such as forming the muscle or functional such as producing enzymes (Keedwell & Narayanan, 2005). In fact, unraveling the function of these proteins has been, and remains, one of the most fundamental problems in biological and clinical research.

A discipline that helps solving such problems is called bioinformatics, a marriage between computer systems and biological ones. In fact, it is the application of computer science to molecular biology in order to analyze and interpret biological data (Zhang & Liu, 2013). These data are produced at high speed.

Currently, 1512 biological databases are indexed in the Molecular Biology Database Collection part of which are published in the 2013 database issue of the journal Nucleic Acid Research (Wittig et al., 2014). Up to date, the National Center of Biotechnology servers incorporate 117 million entries of protein sequences and structures.

Artificial intelligence and data mining techniques are fundamental to analyze and organize fast growing biological data. Indeed, there are various approaches for assigning functional annotations to proteins but there is not a reliable way to do it in a high-throughput manner (Fornes et al., 2014).

Accordingly, in this chapter we will recall basic concepts and works related to proteins functional annotation. It is organized as follows: Section 1.2 describes some basic con-

cepts. Section 1.3 summarizes present approaches concerning proteins function prediction. Section 1.4 provides a brief overview of protein binding cavities.

1.2 Basic concepts

1.2.1 Bioinformatics

Bioinformatics first appeared after the automation of protein and DNA sequencing became possible in the mid-1970s, and after computers became central repositories accessible remotely (Agarwal, 2008).

Bioinformatics play key roles in the identification and prediction of gene functions, the comparison and structural analysis of genomics and the sequencing and discovery of evolutionary relationships between biomolecules. Its applications include pharmaceutical industry for drug discovery and design, analysis of inherited metabolic disorders and the design and configuration of systems analysis treatments (Chen & Hofestadt, 2006).

Bioinformatics is mainly concerned with two components (i) providing software tools and algorithms for large-scale computational analysis and (ii) organization, management and interpretation of biological data (Zhang & Liu, 2013).

There has been a growing interest in organizing proteins into functional modules. In fact, bioinformatics play a crucial role in the creation and application of database resources for protein sequences, structures and interaction networks (e.g. the Protein Data Bank). Moreover, it provides tools to analyze these biological data and interpret the results. For example, having sequenced a new protein, the aim is to compare it with already annotated sequences.

1.2.2 Proteome and proteomics

The proteome is the entire set of proteins species encoded by a certain cell or tissue. Proteomics is a large-scale comprehensive study which investigates the proteome for understanding cellular behavior. It aims to determine sequences and functions of proteins, detect dynamics of protein components, construct protein expression maps and explore protein interacting partners (Maiolica et al., 2012).

Proteomics makes use of technologies such as gel electrophoresis, mass spectrometry, nuclear magnetic resonance, X-ray crytallography and bioinformatics analysis (Zhang &

Liu, 2013). Several structural proteomic projects are producing a growing number of uncharacterized protein structures. Thus, one of the major goals of proteomics is the identification of proteins in biological modules, there is a need for methods that can reliably predict protein functions.

1.2.3 Proteins

Proteins are large molecules composed of chains of "amino acids". Each amino acid has a particular chemical property (e.g. Donor (DO), Acceptor (AC), Mixed donor/acceptor (DA), aromatic (PI) and Hydrophobic aliphatic (AL)) depending on the "R" group which can be variant (e.g. CH_2, CH_2-NH, $CH_3-CH-CH_3$, etc.), they contain 2 other parts, an amino group (generally NH_2) and a carboxyl group (COO^-) as shown in Figure 1.1. There are 20 common amino acids (Agarwal, 2008). A protein may be composed by 50 to 5,000 amino acids arranged side by side.

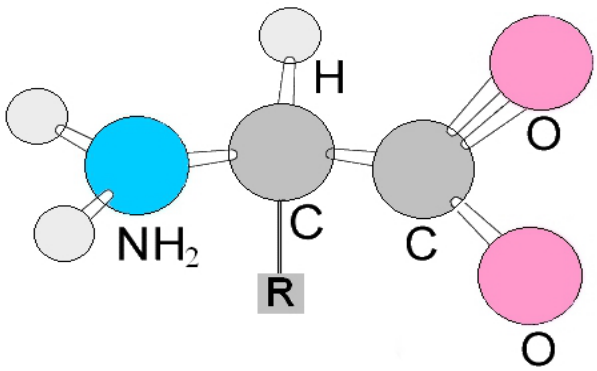

Figure 1.1: Amino acid parts

Proteins form the second largest fraction of cellular weight after water since they perform the majority of cellular functions. They are essential components of cell life as they play many roles such as gene expression, enzymatic reactions, cellular communications and immune reactions. In fact, 39.54% of the genome is composed of protein-coding genes (Niu & Jiang, 2013). Each protein has a specific function which is the cellular role it is committed in. We can summarize some of them in Table 1.1 (SeeKiong, 2004).

Protein families: A protein family is a set of proteins sharing an evolutionary origin. A family may share sequence or structure similarities and therefore exhibit comparable behavior. The organization of proteins into families is gaining worldwide attention. Yet,

Table 1.1: Proteins primary functions

Protein	Function
Enzymes	Control the rate of chemical reactions in cells to facilitate the digestion of food
Contractile proteins (i.e. actin and myosin)	Responsible for all muscular movements
Transport Proteins (i.e. hemoglobin and cytochromes)	Carry and transport molecules between different parts of our body
Hormonal Proteins (i.e.oxytocin, and insulin)	Detect messenger molecules to help coordinate and transmit signals to proteins on the inside of the cells
Storage Proteins (i.e. ovalbumin and casein)	Store amino acids
Structural Proteins (i.e. collagen and keratin)	Provide protection and support

biological properties of proteins are elucidated using costly and time consuming in vitro methods. Bioinformatics treat such problem by use of computational methods. In order to establish protein families on a large scale data, we make use of algorithms based on a notion of similarity.

1.2.4 Protein structure

Proteins may be observed under four structures: primary, secondary, tertiary and quaternary. Coming is a brief description of these structure levels, which are also illustrated graphically in Figure 1.2 from (Campbell & Reece, 2008):

Primary structure: Linear sequence of amino acids joined by peptide bonds.

Secondary structure: Spatially close amino acids bound in three-dimensional space leading to the foundation of substructures as α-helices and β-sheets. This process is called protein folding. The collection of these substructures forms the secondary structure of a protein.

Tertiary structure: Folding of the polypeptide chain in space yield to a nearly stable three-dimensional structure of the protein, this is related to attractive and repulsive

forces among the substructures caused by the folding and depend on the protein function. This folding process leads to the establishment of cavities at the protein surface, some of them enable the binding of the protein with other molecules. These cavities are known as binding sites, there has been a growing interest in exploiting their 3D information. In fact, the molecular function occurs at the surface of the protein as it binds to other molecules causing its reaction. Binding sites are complementary in aspect and physicochemical properties with the binding molecules. Thus, the study of these cavities can reveal functional annotation.

Quaternary structure: Assembling of two or more polypeptide chains producing a set of tertiary structures, the set forms the quaternary structure of the protein.

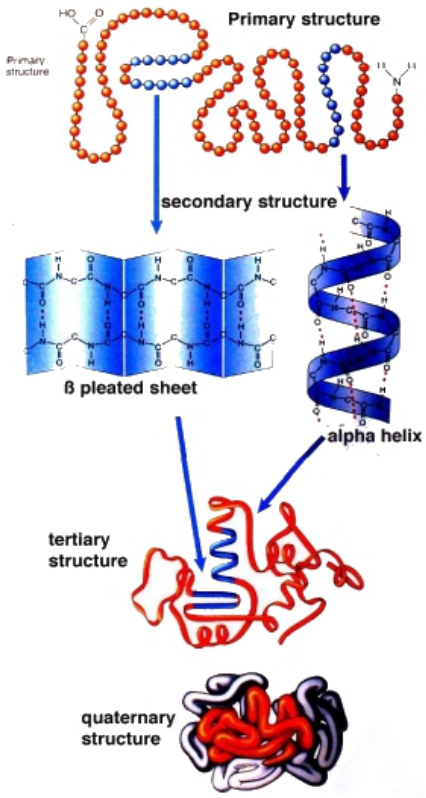

Figure 1.2: Four protein structures

The structural shape of the protein enables it to better perform its function. Let us take the example of hemoglobin which is a folded and compact globular protein. Its navigation through blood vessels is facilitated due to its spherical shape (Skolnick et al., 2000).

In general, the comparison of protein structure is more reliable than the one of sequences, especially, the 3D structure because it is highly conserved among the protein

evolution. In fact, there are cases in which protein belonging to the same families appear distinct when evaluated by sequence comparison alone. We can take the example of the helical cytokines which form an extended family with comparable receptors and signaling pathways, but their similarity is not detectable by sequence comparison and it is only by structure comparison that their shared lineage is conspicuous (Gibrat et al., 1996).

1.2.5 Structure data sources

More and more protein structures are being discovered, several databases are available for storing and treating this structure data such as: PDB (Berman et al., 2000), SCOP (Andreeva et al., 2004) and CATH (Orengo et al., 2002):

- PDB (Protein Data Bank): PDB (Berman et al., 2000) is a popular repository of experimental identification of protein 3D structures. Until today, the experimental methods are X-ray crystallization, fiber diffraction, NMR spectroscopy, electron microscopy, NMR solid-state, neutron diffraction, electron crystallography, solution Scattering and 24 others, for about 84, 955 proteins and peptides. The majority of structures presented in the PDB archive are resulted from X-ray crystallographic experiments (80.936 entries) (Berman, 2013). Besides, there are 6806 structures of other large molecules such as nucleic acids, protein complexes and many others.
- SCOP (Structural Classification of Proteins): SCOP (Andreeva et al., 2004) is a database that adds relevant functional information to data present in the PDB. It aims to provide a description of the structural and evolutionary relationships between known proteins. Recently, the developing lab (Berkeley Lab and UC Berkeley) has settled a new methodology to classify new structures automatically that is being refined. SCOP provides detailed information about proteins whose structure is known and their close relatives, and a framework for future research. SCOP has become a very important standard for techniques of homology detection.
- CATH (Class, Architecture, Topology and Homologous): CATH (Orengo et al., 2002) is a protein structure classification database aiming at organizing PDB structures according to their similarities. Lately, 16 million protein domains were classified into 2,626 superfamilies.

The mentioned databases constitute the core of data sources for structural bioinformatics studies. Additional specialized databases are also available, such as DALI for the comparison of 3D structure of proteins present in the PDB or FSSP for the classification of protein functions based on structure alignment (Jones & Hadley, 1999).

1.3 Major approaches for predicting protein functions

Since knowledge of protein functions is a crucial piece in many fields such as pharmaceutical industry for the development of new drugs (Pérot et al., 2010), agriculture for increasing the quality of crops or in biotechnology advances for the improvement of synthetic biochemicals such as biofuels (Jang et al., 2012). Protein function prediction becomes a challenging task and one of the most important research area in molecular modeling. Existing methods can be sorted into three main categories: sequence-based approaches, structure-based approaches and protein interactions-based approaches (Fornes et al., 2014). In the following, we present a brief description of these methods and some related works.

1.3.1 Sequence-based methods

Sequence comparison is usually used to infer structural and functional conservation, as well as evolutionary relationships of protein families. Generally, sequence-based approaches use the assumption that sequence resemblance entails functional similarity, they transfer characterizations from known proteins to a homologue with unknown function. These methods make profit of clustering, alignment, gene ontology annotations and evolutionary traces. There are methods that use the secondary structure and others that are based on the tertiary structure to infer the protein sequence. However, prediction of tertiary structures from protein sequences is not always reliable since comparable structures are not always available for homologue modeling. On the other hand, secondary structures may be efficiently predicted from sequences (Rost et al., 2003) and can be used to complement sequence homology for function prediction (Ferre et al., 2004).

The simplest way to elucidate a protein characteristics is to use an alignment which is a classical methodology in sequence homology. Alignment softwares such as the BLAST (Basic Local Alignment Search Tool) (Altschul et al., 1990), PSI-BLAST (Altschul et al., 1997) and FASTA (Pearson & Lipman, 1988) are used to find likely homologs by searching in sequence databases or by identifying local duplications within a sequence. They can be used to infer functional and evolutionary relationships between sequences and to identify members of gene families.

Sequence homology presents an effective way for the elucidation of novel protein functions. However, its applicability is limited, especially in cases where no known proteins with similar sequences are found. Moreover, the effectiveness of such a method requires sufficient similarity (above 20% sequence identity) to validate functions inferred from sequences.

Hence, when sequence similarity is not available, we can use similarity of other information. Sequence is not the only aspect, characteristics such as structure can reveal the functional behavior of proteins at a high-throughput manner (Rost, 1999).

1.3.2 Structure-based methods

Protein structure analysis is one of the momentous problems of computational biology because structure is more close to function than sequence. We can compare a protein structure with other known protein structures in order to infer its function. As a matter of fact, structural motifs may predict similar biological functions, help understand protein folding process and is used in building blocks as well as sequence patterns.

Structure-based methods generally use structure information and are often more reliable than sequence-based tools because structures are better preserved than sequences, and proteins with slightly or no sequence resemblance may have similar structures (Klebe et al., 2007). The comparison is mostly done using the three-dimensional structure because it is more conserved in protein evolution and allows to make a link between protein families that appear different when analyzed just by sequence.

Several approaches have proposed structural features in order to infer a protein function, we can classify them into three groups:

- Similarity-based approaches: The function of an unannotated protein is inferred by comparing its structure with other proteins structures whose function is known (Nisius et al., 2012). For example, we can compare binding sites of a particular unknown protein with ones of an annotated protein (e.g., Weskamp et al., 2004, 2007; Klebe et al., 2007; Boukhris et al., 2009). The aim is to find an appropriate representation and a reliable similarity measure. The method described in this master thesis relies on this idea.

- Feature-based approaches: This type of methods identifies three dimensional features conserved among related proteins, then evaluate the difference between the protein function and its structural motifs (e.g., Guruprasad et al., 2000; Chakrabarti et al., 2003). These features are the protein's three dimensional substructure that occur for several related ones. This is used later to predict the function of an uncharacterized protein.

- Learning-based approaches: These approaches (e.g., Fober et al., 2009; Alvareza & Yanb, 2012) use machine learning methods such as classification to determine the

protein function from its structural features, techniques like support vector machines (SVM), neural networks and k-nearest neighbor are employed.

Many structure alignment methods are used for the comparison such as DALI (Holm & Sander, 1995) and VAST (Gibrat et al., 1996). Additionally, machine learning algorithms have shown their effectiveness regarding protein function prediction. It is about training classifiers on a set of labeled structures in order to determine the characterization of an unlabeled entry. Moreover, data mining techniques such as association rules have also been used to identify patterns in a particular secondary structure. (Scott et al., 2011) applied the Apriori algorithm to extract hydrophobicity patterns that achieve a particular secondary structure from a database containing 20,000 proteins extracted from the PDB. They called these motifs "secondary structure formation rules". Their method showed good results in investigating these motifs as well as identifying candidates to structure patterns.

Other approaches use structural similarity analysis of binding pockets to infer a protein function (e.g., Weskamp et al., 2007; Huang, 2009; Boukhris et al., 2009; Nisius et al., 2012), justify and predict cross-reactivities (reaction between different molecules) and study side effects of known drugs. These approaches can complement sequence-based methods and those that use information about folding.

There are some servers that provide binding sites prediction, analysis and functional classification such as FINDSITE (Brylinski & Skolnick, 2008), fPOP (Yuan et al., 2010) and SiteComp (Yingjie et al., 2012). As well as databases such as eF-site[1] which is a database for the storage of protein surfaces and physicochemical properties data or CavBase[2] which is a database used to identify putative protein binding sites from crystal structure data obtained from the PDB (Schmitt et al., 2002). It also provides a representation of the geometry of the binding sites using pseudocenters or dummy atoms which are spatial points exhibiting the physicochemical properties of a surface patch. Each pseudocenter is exhibited as a sphere with a specific color depending on the physicochemical property which is exposed in a surface patch as illustrated in Figure 1.3 from (Schmitt et al., 2002).

Structure-based methods usually make use of 3D protein structures because it offers greater insights into protein functional mechanism. Unfortunately, most tertiary structures are derived using costly and time consuming experimental methods such as X-ray crystallography (about 90%) and magnetic resonance spectroscopy (NMR) (about 9%). Currently, coverage of tertiary structures is relatively low which limits their use in function prediction.

[1] eF-site
[2] CavBase

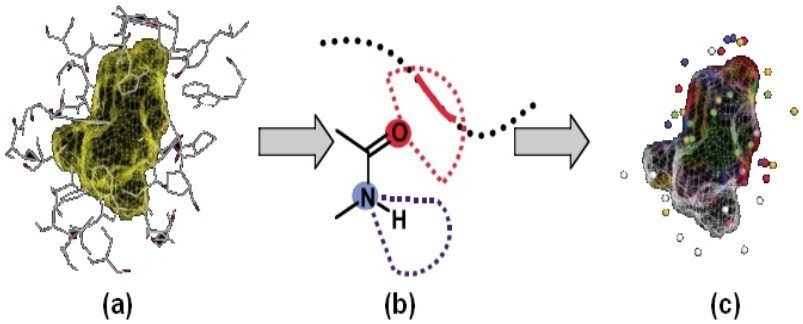

Figure 1.3: Protein binding sites representation within CavBase

1.3.3 Protein interactions-based methods

Protein-protein interactions refer to associations between protein molecules, which include direct physical binding and genetic interactions. According to (Chua et al., 2007), protein-protein interactions analysis is a complementary approach to sequence homology for the inference of the function of the protein. Interaction of proteins plays an important role in inter cellular communication, in signal transduction and in the regulation of gene expression.

Proteins interact with other biological molecules such as DNA, RNA, and other proteins. We can infer protein function by analyzing its relationships with other proteins because they interact so as to undertake their biological functions. Generally, a protein shares cellular links with other proteins forming a network. A common way of visualizing this network is as graphs, where nodes represent several proteins and edges represent their interactions (Li et al., 2005). Such a representation can enable researchers to infer characteristics of proteins from those of proteins that are not directly interacting with them.

In general, we can categorize protein function prediction methods using protein-protein interactions into two groups. The first group, referred to as local prediction methods, elucidates a protein function based on its neighbors in local interaction neighborhood such as the neighbor counting method (Schwikowski et al., 2000) which assigns neighbor counting scoring function for the most frequent protein among its interaction partners and assigns the function of the one having the largest score. The second group, referred to as global optimization techniques, infers each protein function in a way that the final annotations for all proteins composing the network are optimal considering some measure. This group of methods seeks to optimize an objective function. Examples include the Simulated Annealing (Vazquez et al., 2003) method which is similar to based neighbor

counting, the main concept is that proteins that interact together should share the same function. However those that are not interacting should not.

Other methods consider the problem of function prediction in protein-protein networks as a clustering problem. The aim is to find a cluster of nodes of a graph such that there are many edges within the same cluster but very little edges between two different clusters.

Nevertheless, some researches have shown that, in some cases, a protein does not share functional activities with any of its interaction partners. Instead, it shares some functions with a protein sharing interactions partners with it. This observation is biologically significant because docking to a particular binding site of a common neighbor require for the two proteins to have similar biochemical properties (Gursoy et al., 2008). Hence, this is difficult to prove, since similar interacting partners may work at different times or in different pathways.

1.4 Protein binding sites

The interacting sites (residues, cavities) among the protein surface are of interest in proteome research. Protein binding sites (pockets) are small cavities at the surface of the protein that bind with other molecules as shown in Figure 1.4 from (Nisius et al., 2012) and cause its reaction. They vary in size and shape, coming in a nearly spherical shape or forming a curved chip (Pérot et al., 2010).

1.4.1 Binding sites presentation

Proteins interact with plenty of other molecules such as: DNA, RNA, other proteins and small molecules. Complementarity in shape and chemical properties are the basis of molecular recognition, the identification of the binding sites can provide perception into the protein function since the molecular activity take place at their level, they are chemically and physically complementary to the binding molecules (Weskamp et al., 2007). Lately, binding pockets have had new emerging perceptions such as the notion of "druggabale" pockets and centric chemical space binding site (Pérot et al., 2010). Accordingly, similarity analysis of binding sites can provide solutions to pharmaceutical discoveries as the binding process is similar to the mechanism of action of drugs (Fuller et al., 2009). We can take the example of the SOIPPA approach which appointed similarities between binding pockets of human catechol-*O*-methyltransferase (COMT), a target in Parkinson's disease, and enoyl-acyl of the protein reductase (InhA), a tuberculosis target. Therefore, it was

Figure 1.4: 3D visualization of the endonuclease (an enzyme) and its binding sites (highlighted in blue)

suggested that two marketed drugs (entacapone and tolcapone) that inhibit COMT may inhibit InhA and thus may treat tuberculosis (Kinnings et al., 2009). Moreover, Binding sites similarities recognized between irrelevant proteins may help to resolve and predict cross-reactivities (Klebe et al., 2007). In this respect, discovery of these cavities is a key for understanding mechanisms of protein interactions.

1.4.2 Binding sites detection

Several methods have been developed for the detection of binding pockets available via web-services such as QSite-Finder, SCREEN, CASTp, MetaPocket, PocketDepth and Fpocket (Huang, 2009). We can use three different types of data sources in order to infer the location of lingand binding cavities at the protein surface: protein structure, evolutionary information (sequence alignments) and ligand information.

1.4.3 Protein function prediction using binding sites

Binding pockets are important contributors in the functional process of structures. Various approaches are proposed to infer an unknown function of a protein by comparing its binding ligand structure with another of a known protein function then computing their similarities.

The result may indicate functional resemblances and hence predict the function of the respective biomolecule. One way to accomplish this task is to represent binding pockets by use of graphs. Indeed, graphs are a suitable form to represent structured objects. We model the geometrical taxonomy and the physicochemical properties of the proteins binding sites by pseudocenters described by the graph nodes.

Approaches analyzing binding cavities may be divided into three categories: (i) methods that extract and match cavities geometrically, (ii) methods that detect and compare particular geometrical patterns of amino acids in binding sites and (iii) methods that use evolutionary motifs to predict binding sites locations. In short, all these methods suggest comparison of atomic correlation in one way or another (Kahraman et al., 2007).

One of the methods of the first category is Cavbase, a database that enables the extraction and storage of binding sites from experimentally determined protein structures. It uses the LIGSITE algorithm for the detection of binding pockets on the protein surface. The identification process follows a series of operations on a cubic grid and shows good precision results during an efficient time (Hendlich et al., 1997). Binding sites are stored according to the cavity-flanking amino acids characterizing their physicochemical properties, each represented by pseudospheres which are 3D descriptors coding the molecular recognition (Schmitt et al., 2002).

The possible types used in CavBase are: Donor (DO), Acceptor (A), Mixed donor/acceptor (DA), $\pi - \pi$ interaction (PI), hydrophobic aliphatic (AL), aromatic (Ar) and Ion metal (IM) (Schmitt et al., 2002). These physicochemical properties account for different types of possible interactions between residues of the binding site and the substrate of the protein.

1.4.4 Binding sites representation

Protein binding sites are usually modeled using a simple, connected, undirected, node-labeled and edge-weighted graph. Physicochemical properties $l(v) = \{1,\ldots,7\}$, (DO, A, DA, PI, AL, IM and Ar) (Schmitt et al., 2002) are interpreted as nodes labels, and edges represent links between two centers. In fact, two pseudospheres are linked if the Euclidean distance separating them is below 11Å [3], actually, it is considered sufficient to capture the binding site geometry, thus longer edges are ignored (Weskamp et al., 2007). This distance is expressed by the edges weights, it is used in order to reduce the complexity of the graph representation and restrain node connections. Therefore, graphs enable structure conservation.

[3] Angström: unit of height of molecules = 10^{-10} m

1.5 Conclusion

In this chapter, we have described some related works about protein function prediction. In fact, it is one of the crucial problems of bioinfomatics which is the application of information technology to manage and analyze biological data. Algorithms and tools from artificial intelligence and data mining techniques are used to treat, organize and analyze this incredibly fast generated data. Sequence alignment, protein interactions and structural comparison of proteins are different ways to predict protein functions. Indeed, these techniques are often carried out by automated methods such as machine-learning algorithms. We can classify protein functions according to their binding cavities.

One of the existing methods consists in representing binding pockets using graphs then compare them via a similarity measure. Approaches performing this task use graph matching techniques which will be presented and illustrated in the following chapter.

Chapter 2

Graph similarity

2.1 Introduction

Graphs are powerful tools to represent connections and relationships between objects. They have been very useful as models in many areas such as marketing, economics, engineering, biology, pattern recognition, social networks, computer vision and computer aided design.

In particular, graph-based models have been successfully used in computational structural biology, to better understand complex systems. We can model protein structure properties as graphs. Therefore, the problem of measuring similarity between different proteins turns into the problem of computing similarity between graphs, known as the graph matching problem from graph theory.

Structured data presented in different servers such as the PDB can be easily transformed into graphs. In fact, a graph is a diagram that represents objects as a set of nodes[1]/vertices[2] and the relationships between them as a set of edges. These edges can express a geometric relation, a neighborhood relation or a temporal link. In order to enhance the graph representation, further information such as labels and attributes are employed to describe nodes and edges. For example, we can add weights to edges to express spatial distances between two linked nodes.

Protein binding sites extracted from protein 3D structures can be modeled using a graph representation where vertices express the binding atoms and edges express links between two connected ones (Weskamp et al., 2007). Therefore, the task of predicting a

[1] Points in 2D space
[2] Points in 3D space

protein function through similarity computation of binding pockets turns to a matching of graphs.

Accordingly, this Chapter is organized as follows: Section 2.2 presents the graph matching problem known from graph theory. Section 2.3 describes protein function prediction using graph representations of binding sites.

2.2 Graph matching

Matching problems arise in several applications such as resource allocation, traffic balancing, pattern recognition, computer vision, etc. A simple graph matching problem consists in searching the maximum number of compatible pairs (edges or nodes) given a graph.

In applications such as molecular biology, graph matching techniques can be used to setup constellation approaches to compute similarities (Alvareza & Yanb, 2012), extract common and discriminative motifs and detect structural variations (Weskamp et al., 2007). The invariance and versatility properties of graphs made them more and more important in bioinformatics where structure similarity is an essential issue. As a matter of fact, if we alter, rotate or translate a graph to its mirror image, it will still the same in a mathematical sense (Bunke, 2000).

We can categorize the graph matching problem into two groups (i) Exact graph matching and (ii) inexact graph matching. Exact matching includes graph/subgraph isomorphism, whereas, inexact graph matching is an error tolerant graph matching that uses graph edit distance.

2.2.1 Exact matching

We compare graphs in order to measure resemblance, identify a structure over a set of already known structures or to detect discriminative motifs. Exact matching is used for determining whether two objects are the same. It includes graph and subgraph isomorphism which are described in the following as well as their complexity.

2.2.1.1 Graph isomorphism

The simplest form of exact graph matching is graph isomorphism (GI) which is used in order to check whether two graphs are identical or not. It's a correspondence based on the

following structural characteristics:

1. Nodes labels in the two graphs must be the same.
2. An edge exists if it is present between the corresponding matched nodes in the two graphs.
3. Edges labels in the two graphs must match each other.

Given two graphs $G = (V, E, l_V, l_E)$ and $G' = (V', E', l'_V, l'_E)$, where $|V|$ is the set of vertices, $|E|$ is the set of edges, l_V is a function that assigns labels to vertices and l_E is a function assigning labels to edges. A graph isomorphism G and G' is a bijective mapping $f : V \rightarrow V'$ conserving all the labels and the structure of the edges such that, for all $u, v \in V$, and $(u, v) \in E$:

1. $l_V(u) = l\hat{a}^2{}_V(f(u)), l_V(v) = l\hat{a}^2{}_V(f(v))$
2. $(f(u), f(v)) \in E'$
3. $l_E(u, v) = l\hat{a}^2{}_E(f(u), f(v))$

If such a mapping exists, then G and G' are called isomorphic and are denoted $G \approx G'$ (Ferrer et al., 2009).

Several algorithms are proposed for isomorphism testing of graphs. One of the best algorithms is "Nauty" proposed by Brendan McKay (McKay, 1981). Nauty creates a canonical labelling of the graph vertices and searches generators of the automorphism[3] group in order to prune the search space. Hence, it is fast for most graphs but it has exponential complexity for certain graph classes. Other efficient algorithms for testing whether two graphs are isomorphic include Conauto (Lopez-Presa & Anta, 2011) which consists in finding compatible sequences of partitions for the two graphs.

Graph isomorphism is considered to be an NP problem. There is no polynomial algorithm that is known and it is questionable whether it is in co-NP or to be NP-complete. However, NP-completeness is considered improbable since it would entail collapse of the the polynomial-time hierarchy (McKay & Piperno, 2014). Besides, it is solvable in polynomial time for a certain graph classes such as trees or planar graphs.

[3] An isomorphism from a graph to itself

2.2.1.2 Subgraph isomorphism

We can use subgraph isomorphism if we want to find whether an object is part of another one or it is present in a group of objects. It has been applied in many fields such as image processing, computer vision, pattern recognition, biocomputing and cheminformatics. Accordingly, subgraph isomorphism is about finding a subgraph G_{01} of G_1 which is isomorphic to a graph G_2.

There are various algorithms proposed for this type of problem. One of the well-known algorithms is the one developed by (Ullmann, 1976). It uses back-tracking in order to eliminate nodes from the search space.

Subgraph isomorphism is known to be an NP-complete problem. It is often reformulated to a decision problem where the solution is positive if a graph G_1 is isomorphic to a subgraph of G_2 otherwise, it is negative. For some specific classes of graphs, subgraph isomorphism has a lower complexity. For instance, in the case where G_1 is a forest and G_2 is the matched tree, the problem has been shown to be of a polynomial complexity (Garey & Johnson, 1979).

2.2.1.3 Maximum common subgraph

The basis of graph matching is to seek for the maximum common subgraph (*mcs*). We can use *mcs* to measure the similarity between two graphs by searching for the maximum common subgraph isomorphism.

Formally, given a graph $G = (V, E)$, a subset $V_S \subseteq V$ of its vertices and a subset $E_S \subseteq E \cap (V_S \times V_S)$ of its edges. $G_S(V_S, E_S)$ is called the subgraph induced by V_S if $V_S \subseteq V$ and $E_S = E \cap (V_S \times V_S)$. Therefore, $G_{cs} = (V_{cs}, E_{cs})$ is a common subgraph of two graphs $G = (V, E)$ and $G' = (V', E')$ if $G_{cs} \approx G_S$ and $G_{cs} \approx G'_S$ where G_S and G'_S are respectively two induced subgraphs of G and G'. A common subgraph is maximum if there is no other common subgraph of G and G' with more nodes as illustrated in Example 2.1 Usually, the *mcs* is not unique.

Example 2.1. *In this Example, two graphs G (a) and G' (b) have a subgraph isomorphism (c) composed of nodes having the same color labels, it is also a mcs since no other common node can be added to it (see Figure 2.1).*

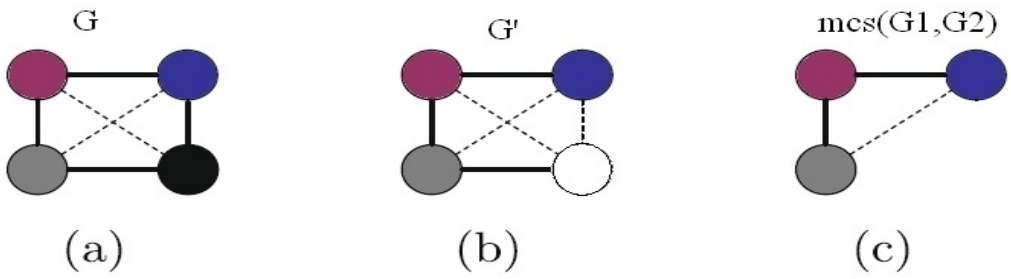

Figure 2.1: A possible *mcs* (c) of two graphs (a) and (b)

Several methods measure similarity of two graphs by computing the size of the *mcs* in terms of number of nodes as illustrated in Equation 2.1 (Horst & Kim, 1998).

$$\delta(G_1, G_2) = \frac{|mcs(G_1, G_2)|}{max(|G_1|, |G_1|)} \tag{2.1}$$

Where $|mcs(G_1, G_2)|$ = number of nodes of the *mcs*, and $max(|G_1|, |G_1|)$: maximum number of nodes between the two graphs. Clearly, the larger the maximum common subgraph is, the more similar the two graphs are. Similarly, maximum common subgraph can solve the computation of the minimum common supergraph (Bunke et al., 2000). Formally, G_3 is a supergraph of G_1 and G_2, if there is a subgraph isomorphism from G_1 and G_2 to G_3 (Bunke, 2000). G_3 is a minimum common supergraph (MCS) if there is no other supergraph with less nodes as illustrated in Figure 2.2.

Example 2.2. *Let us consider the three graphs G_1 (a), G_2 (b) and G_3 (c) depicted in Figure 2.2, a supergraph G_4 (d) is a graph composed of all the nodes and the possible connections shared between them, G_4 is also a minimum common supergraph.*

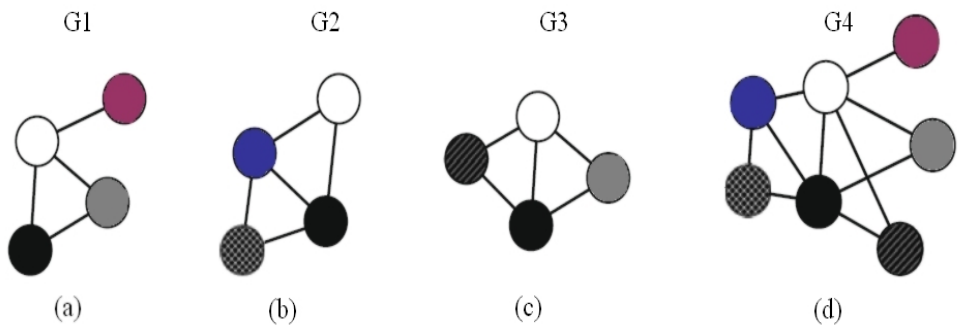

Figure 2.2: A minimum common supergraph (d) for a set of three graphs (a), (b) and (c)

We can compute the similarity between two input graphs using maximum common subraph (mcs) and minimum common supergraph (MCS). Equations 2.2 and 2.3 are metrics used to measure the similarity between two graphs G_1 and G_2.

$$\delta_1(G_1, G_2) = 1 - MCS(G_1, G_2) \times max(|G_1|, |G_2|) \tag{2.2}$$

$$\delta_2(G_1, G_2) = mcs(G_1, G_2) - MCS(G_1, G_2) \tag{2.3}$$

Seeking for the mcs is an intractable NP-complete problem (Bunke, 2000), and no exact algorithms have been discovered. Although, it can be reduced to the problem of determining the maximum clique in an association graph. This reduction is explained in the following.

2.2.1.4 Cliques

We can reduce the problem of finding a common subgraph to a similar problem consisting in searching for cliques. As a matter of fact, a clique is a subset of vertices all connected to each other. Therefore, a common subraph that is completely connected forms a clique. A maximum clique in a graph is a clique having maximum number of nodes and its size is called the clique number of the graph.

There are several algorithms proposed for detecting cliques and maximum cliques. One of the fastest algorithms is the one proposed by Joep Kerbosch and Coenraad Bron, the Bron-Kerbosch algorithm (Bron & Kerbosch, 1973). It is very efficient in most of the cases for enumerating all maximal cliques. The main idea behind it is to use a backtracking method based on an heuristic truncation of a search tree, i.e. if a clique q composed of k vertices v, $q = v_1 \dots v_k$, is found, than we search for the set of vertices that may extend q (vertices that come after the last vertex of the set of vertices composing q), also known as candidate extensions that we denote C, $C = \tau_{v_1} \cap \dots \cap \tau_{v_k}$, where τ_{v_i} is the candidate extension of the vertex v_i, if $C = \emptyset$, than q is maximal.

The reduction of the mcs problem to the clique problem is done by use of a compatibility graph also called an association graph or product graph. This graph has the property that a maximum common subgraph between two compared graphs is equivalent to a maximum clique.

Formally, a product graph $G_P = (V_P, E_P)$ of two graphs $G = (V, E)$ and $G' = (V', E')$ is defined by a set of nodes: $V_P \subseteq V \times V'$ and a set of edges: $E_P \subseteq V_P \times V_P$ such

as $V_P = \{(v_i, v'_j) \mid l_V(v_i) = l_V(v'_j)\}$ and $E_P = \{((v_i, v'_j), (v_k, v'_l)) \mid l_E(v_i, v_k) = l_E(v'_j, v'_l)\}$. Therefore, finding a maximum common subgraph amounts to finding a maximal clique in G_P.

Example 2.3. *If we consider the two graphs G_1 (a) and G_2 (b) illustrated in Figure 2.3, a product graph G_p (c) will be composed of the nodes having the same labels in G_1 and G_2. As for the connections, an edge exists in G_p if it links two matched nodes in both G_1 and G_2, and it must have the same weight. The mcs of G_1 and G_2 is the maximal clique in G_p, which in our example will be composed of the purple, blue and black nodes.*

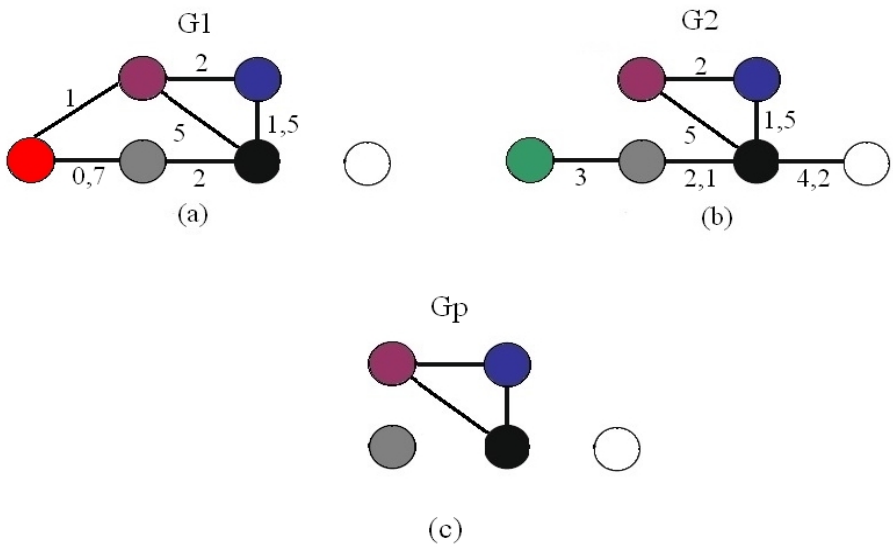

Figure 2.3: A product graph (c) of two graphs (a) and (b)

The task of finding whether a clique of a given size exists in a graph is NP-hard (Garey & Johnson, 1979). Besides, (Håstad, 1997) shows that there cannot be any polynomial algorithm that finds approximately the maximum clique within a factor better than $\mathcal{O}(n^{1-\epsilon})$, for any epsilon $\epsilon > 0$.

2.2.2 Inexact matching using graph edit distance

In real world applications, one cannot expect to find perfect match between two graphs. Thus, we have to take into account the possibility of errors. For example, real world objects can be affected with noise. Therefore, the graph matching process need to integrate some degree of error tolerance. Thus, these errors are quantified using a function known as the

graph edit distance. Graph edit distance is often used to compute dissimilarity between two input graphs. The main idea is to define this dissimilarity as the minimum possible amount of distortion that can be applied to transform a graph G_1 into a graph G_2. Generally, an edit operation or distortion is either an insertion, a deletion or a substitution that can be applied to nodes or edges. We can transform a graph to another one by a sequence of graph edit operations or edit paths.

Usually, costs are assigned for each edit operation in order to evaluate which one is the best. For example, if an edit path i is more likely to occur than it will be assigned by the smallest cost c_i. On the other hand, if an edit operation i generates a big amount of distortion in the transformation, it will be assigned by the higher cost c_i. A graph edit distance is defined by the sequence of operations that has the minimum cost. Accordingly, the shorter (longer) the sequence is the more (less) similar the graphs are.

Formally, the edit distance between two graphs G_1 and G_2: $\delta(G_1, G_2)$ is given by the minimum cost of edit operations transforming one graph into the other.

Example 2.4. *Let us consider the two graphs G_1 (a) and G_2 (b) illustrated in Figure 2.4, the task is to transform G_1 into G_2. A sequence of edit operations (d) includes node and edge deletion (the red node and its relative edges), node and edge insertion (green node and its relative edge) and edge substitution (the edge connecting the black and white nodes). Since each operation is defined by a cost, the edit graph distance is given by the sum of the costs for all operations in the sequence*

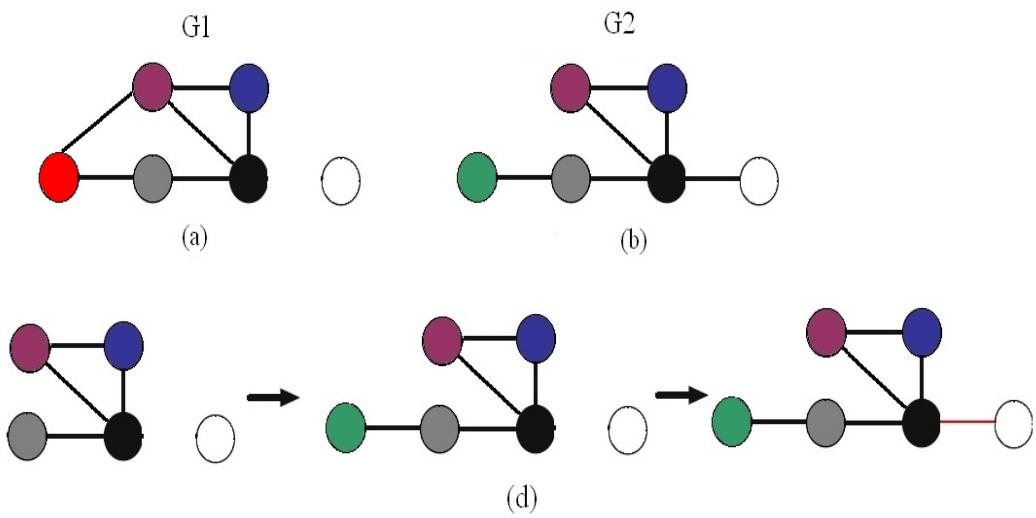

Figure 2.4: An edit path (c) for two graphs (a) and (b)

2.3 Protein function prediction using graph representation of binding sites

Graphs are gaining more and more attention in molecular biology. Several methods are proposed to compare graphs that represent protein structures and compute related similarity measures. In fact, there are two main steps. The first step finds the maximum common subgraph using graph matching. The second step predicts the protein's function by measuring the similarity computed between the two inputted graphs corresponding to a characterized and an uncharacterized proteins.

The goal is to find an appropriate similarity measure on graphs which is tolerant against biological variations, conformational flexibility measurement errors, mutations and large differences in size.

2.3.1 Graph representation of binding sites

A graph G representing protein binding sites can be defined as a collection of two sets V and E and two functions l_V and l_E, $G(V, E, l_V, l_E)$ where:

- $V = \{v_i, i \in [1 \ldots |V|]\}$: is a set of vertices representing binding atoms, where $|V|$ is the number of vertices of G (cardinality).
- $E = \{(v_i, v_j)\} \subset V \times V$: is a set of edges representing connections between binding atoms that are linked in spatial space, where (v_i, v_j) represent an edge between v_i and v_j, and $|E|$ is the number of edges of G.
- $l_V : V \to \mathbb{N}$: is a node labeling function.
- $l_E : V \times V \to \mathbb{R}$: is an edge weighting function.

Wherein, the labeling function $l_V = \{1 \ldots 7\}$ where $1 \ldots 7$ express the physicochemical properties presented in cavbase which we mentioned in Chapter 1 (Donor (DO), Acceptor (A), Mixed donor/acceptor (DA), Ion metal (IM), hydrophobic aliphatic (AL), aromatic (Ar) and $\pi - \pi$ interaction (PI)) and the weighting function l_E describes distances between two connected vertices. As discussed in Chapter 1, two binding atoms are connected if the distance that separates them is below 11Å as illustrated in Figure 2.5.

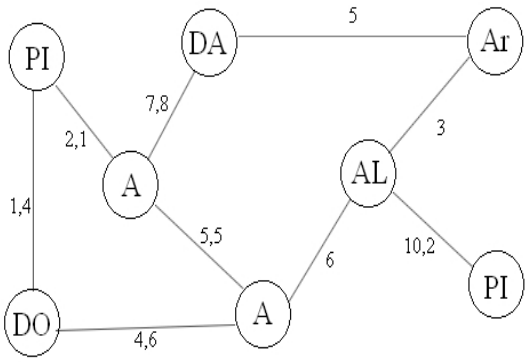

Figure 2.5: Graph representing protein binding sites

2.3.2 Prediction using graph matching

As already mentioned in Chapter 1, more precisely in Section 1.4, the comparison of protein binding pockets may lead to infer a protein function. Indeed, we can compare the binding pockets of a protein whose function is known with the binding pockets of another protein whose function is not known, then the result allows measurement of the similarity between the two proteins and therefore predict if the latter has the same function as the first protein. Since graph matching allows such comparison, we can use it to infer the protein function as illustrated in Figure 2.6.

Several previous works have used similarity measure based on the concept of the maximum common subgraph for comparing chemical compounds (e.g., Gardiner et al., 1997; Raymond & Willett, 2002; Raymond et al., 2002). Unfortunately, when comparing protein binding sites, exact matches are difficult to find as a result of noise in data, mutation or molecular flexibility. In addition, proteins are naturally very flexible thus, 3D-structure may induce distortions. Accordingly, an alteration of the spatial arrangement of the pseudocenters may affect the protein binding pockets and therefore the graph nodes coordinates. Besides, even if we reformulate the *mcs* detection problem into the clique extraction from the product graph, it will still be very restrictive since cliques require that all nodes and edges labels eventually to be equal. Thus, it will result in small common subgraphs unable to capture structural similarities. Besides, maximum common subgraph is a sensitive measure toward deviations and errors especially when we work with graphs extracted from real-world data whose edge weights are real-valued (Boukhris et al., 2009).

One of the methods proposed to solve the problem of exact matches consists in relaxing the exact matches condition by reducing constraints of extracting the product graph and

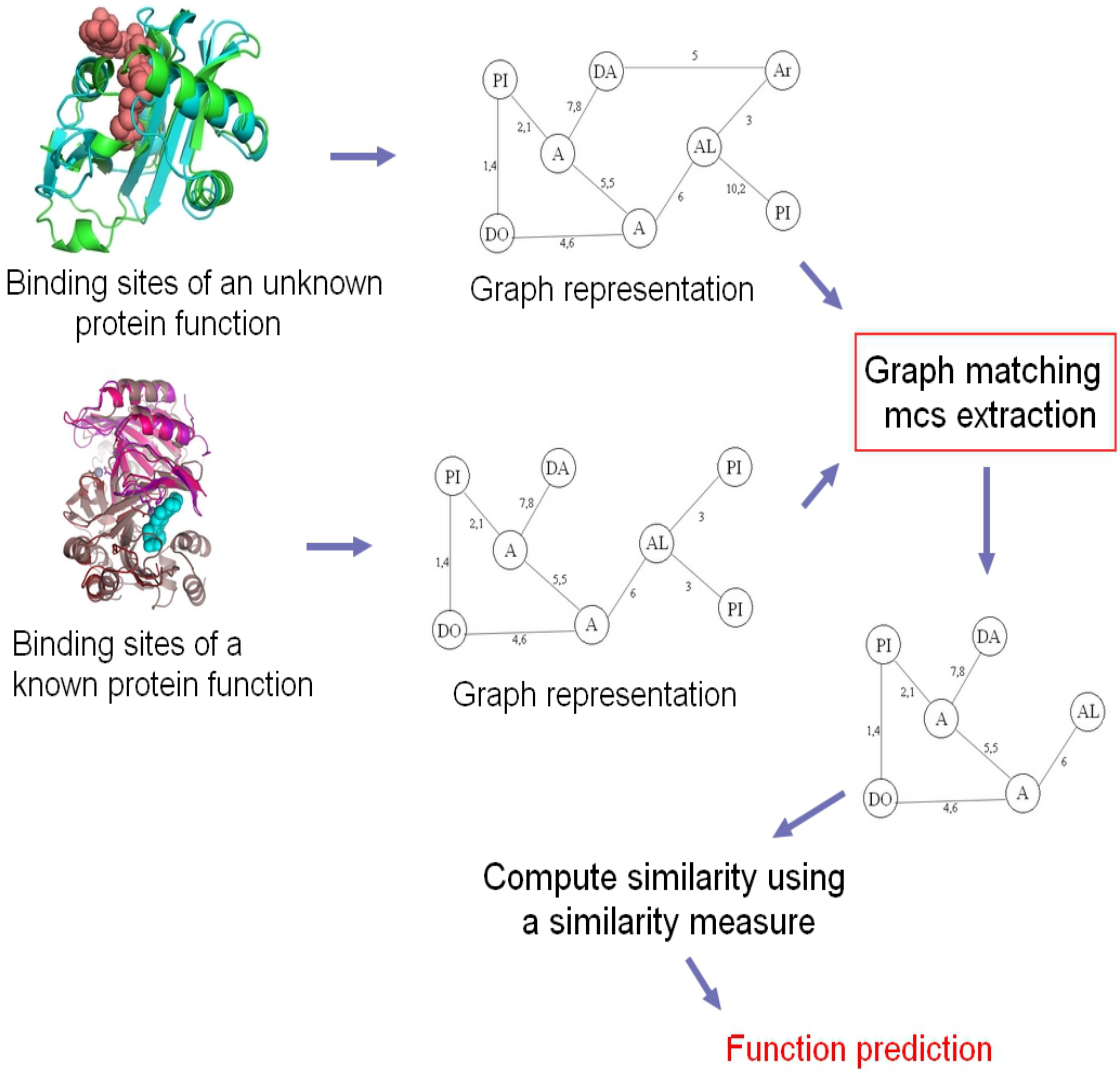

Figure 2.6: Prediction based on binding sites using graph matching

detecting "approximately" common subgraphs. To this end, the concept of quasi-clique is employed which will be presented in the following Subsection.

2.3.3 Quasi-clique

A quasi-clique is an "almost completely" connected graph. To be specific, each vertex in a quasi-clique is not necessarily connected to all the other vertices, instead, it is connected to at least $\gamma \cdot (n-1)$ other vertices, where γ is a relaxation parameter specified by the user

that satisfies $0 \leqslant \gamma \leqslant 1$ and n is the number of vertices of the γ-clique. Thus, the name *quasi* which means almost or "γ-quasi-clique" or simply a "γ-clique" (Pattillo et al., 2013). Another definition is based on the density of edges where a graph $Q(V_Q, E_Q)$ is a γ-clique if its edge set cardinality is $|E_Q| = \gamma \cdot \binom{|V_Q|}{2}$. The problem of maximum γ-quasi-clique detection requires that the γ-clique has the maximum number of vertices.

Example 2.5. *As illustrated in Figure 2.7, a graph composed of a set of vertices* $|V| = 5$ *and a set of edges* $|E| = 5$*, if* $\gamma = 0,5$ *than each vertex must be connected to at least* $0,5 \cdot (|V| - 1) = 0,5 \cdot (5 - 1) = 2$ *other vertices.*

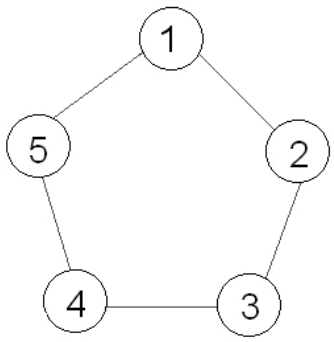

Figure 2.7: 0.5-quasi-clique

Returning to the problem of *mcs* detection explained in Section 2.2, more precisely in Subsection 2.2.1.3, the alternative was to detect the largest clique in the product graph. Indeed, the product graph extraction is also restrictive since an edge is inserted if it has exactly the same weights in the matched edges between two matched nodes. Besides, in order to introduce some tolerance in the product graph G_p , we can define the set of edges as:

$$E_P = \{((v_i, v'_j), (v_k, v'_l))| \parallel l_E(v_i, v_k) - l_E(v'_j, v'_l) \parallel \leqslant \epsilon\} \tag{2.4}$$

Where ϵ is a tolerance threshold parameter for edge height differences specified by the user. Therefore, the problem of finding maximum common subgraphs of two graphs G_1 and G_2 from the product graph amounts to find a maximum γ-quasi-clique.

Detection of the maximum γ-clique in a graph is an NP-complete problem. Since the downward closure property does not hold for quasi-cliques, mining these graphs is a more challenging task. In fact, the downward-closure property means that if a graph G is a clique than all of its induced subgraphs form cliques. Unfortunately, this property is not

applicable for quasi-cliques since an induced subgraph of such a graph may not form a quasi-clique.

Approaches such as (Abello et al., 2002; Liu & Limsoon, 2008; Daxin & Jian, 2009) mine quasi-cliques by pruning the search space of the vertex set. For instance, the method proposed by (Boukhris et al., 2009) (which we call the pruning-based method) is based on the pruning of the search space of nodes to detect the maximum quasi-clique in the context of comparing two graphs of protein binding pockets. Indeed, pruning techniques are applied in order to reduce the search space and therefore decrease the run time complexity. The principle of this method is to explore in a depth-first search the set enumeration tree of all vertices. Since the latter is exponential to the graph size (number of vertices), they applied some pruning techniques from (Liu & Limsoon, 2008) to reduce the search space which are described in the following:

- Minimum size degree: The minimum number of vertices in a quasi-clique fixed in order to eliminate trivial ones.
- Diameter of the graph: the upper bound of the diameter of a γ-quasi-clique is related to the value of γ. This upper bound can be used to reduce the candidate extensions of a vertex set.
- A lookahead technique: This technique is used to check whether a set X forms a γ-quasi-clique by unifying it with its candidate extensions, if so, there is no need to extend X further because $X \cup Cand_exts(X)$ includes all the vertex sets that can be extended from X thus they cannot be maximal.
- The degree of the vertices: The search space is pruned according to the number of neighbors of X based on $indeg^X(u)$ and $exdeg^X(u)$ (where $indeg^X(u)$ is the number of vertices in X that are adjacent to u and $exdeg^X(u)$ is the number of vertices in $Cand_exts(X)$ that are adjacent to u).
  - $indeg^X(u) + exdeg^X(u) < \gamma \cdot (|X| + exdeg^X(u))$
  - $indeg^X(u) < \gamma \cdot |X|$ and $exdegX(u) = 0$
  - $indeg^X(u) + exdeg^X(u) < \gamma \cdot (|X| - 1 + exdeg^X(u))$
- The upper bound U and the lower bound L: The Upper and Lower bounds report respectively to the number of vertices that can be added to X and the number of vertices that can be added to increase the degree of vertices in X to form a γ-quasi-clique.

- Critical vertices: A vertex $u \in X$ is considered critical if $indeg^X(u) + exdeg^X(u) = \gamma \cdot [|X| + L_X - 1]$. If u is critical, then the vertices that are adjacent to u in $Cand_exts(X)$ are added to X.
- Cover vertices: The vertices that are covered by a vertex cover u are removed from $Cand_exts(X)$ (where u is a vertex cover if it maximizes $Cand_exts(X) \cap N_G(u) \cap (\cap_{v \in X \wedge (u,v) \notin E} N_G(v))$ ($N_G(v)$ is the degree of V).

Nevertheless, even with the application of these pruning techniques the complexity remains high since the construction of the set enumeration tree costs $O(2^n)$ where n is the total number of vertices.

Other examples include a recent approach proposed in (Fober et al., 2013) for mining the maximum quasi-clique (which we call the merging-based method) with application to protein binding sites comparison. It is based on the two following steps:

1. Detect local cliques on a graph $G = (V, E)$ as follows:
   - Select a node $v \in V$
   - Derive a neighborhood graph $G(v) = (V_v, E_v)$ defined by $V_v = \{v\} \cup \{v_i \in V : (v, v_i) \in E\}$ and $E_v = E \cap V_v \times V_v$
   - Remove iteratively nodes with smallest degrees in V_v with their adjacent edges until the clique property is satisfied.
2. Merge the local cliques to form maximal quasi-cliques
   - Store the detected local cliques in a set LC and a beam C.
   - Consider all pairs (G_C, G_L) where $G_C \in C$ and $G_L \in LC$. If $|V_C \cap V_L| / min\{|V_C|, |V_L|\}$ is above a threshold w than G_C and G_L are merged into a temporal set. At the same time, store the quasi-clique with the maximum number of nodes with at least a density above γ that is found so far into a variable Q.
   - Replace the beam C by the temporal set and continue until a certain number of iterations is attained.

At the end, Q is considered as the maximum quasi-clique.

The drawback of this method is that the cliques merging has exponential complexity because the size of the beam C becomes very large after few merges, up to $O(2^n)$ (n being the number of vertices).

A summary of the maximum quasi-clique detection on the product graph is illustrated in Example 2.6.

Example 2.6. *Let G (a) and G' (b) be two graphs representing two protein binding sites illustrated in Figure 2.8. The vertices' labels symbolize the physicochemical properties where PI, A, Al and DA stand respectively to $\pi - \pi$ interaction, Acceptor, Hydrophobic Aliphatic and Donor/Acceptor. The edges' weights express the distances between these nodes. In order to ease the graph comprehension, numbers are added near to each vertex in Figure 2.8.*

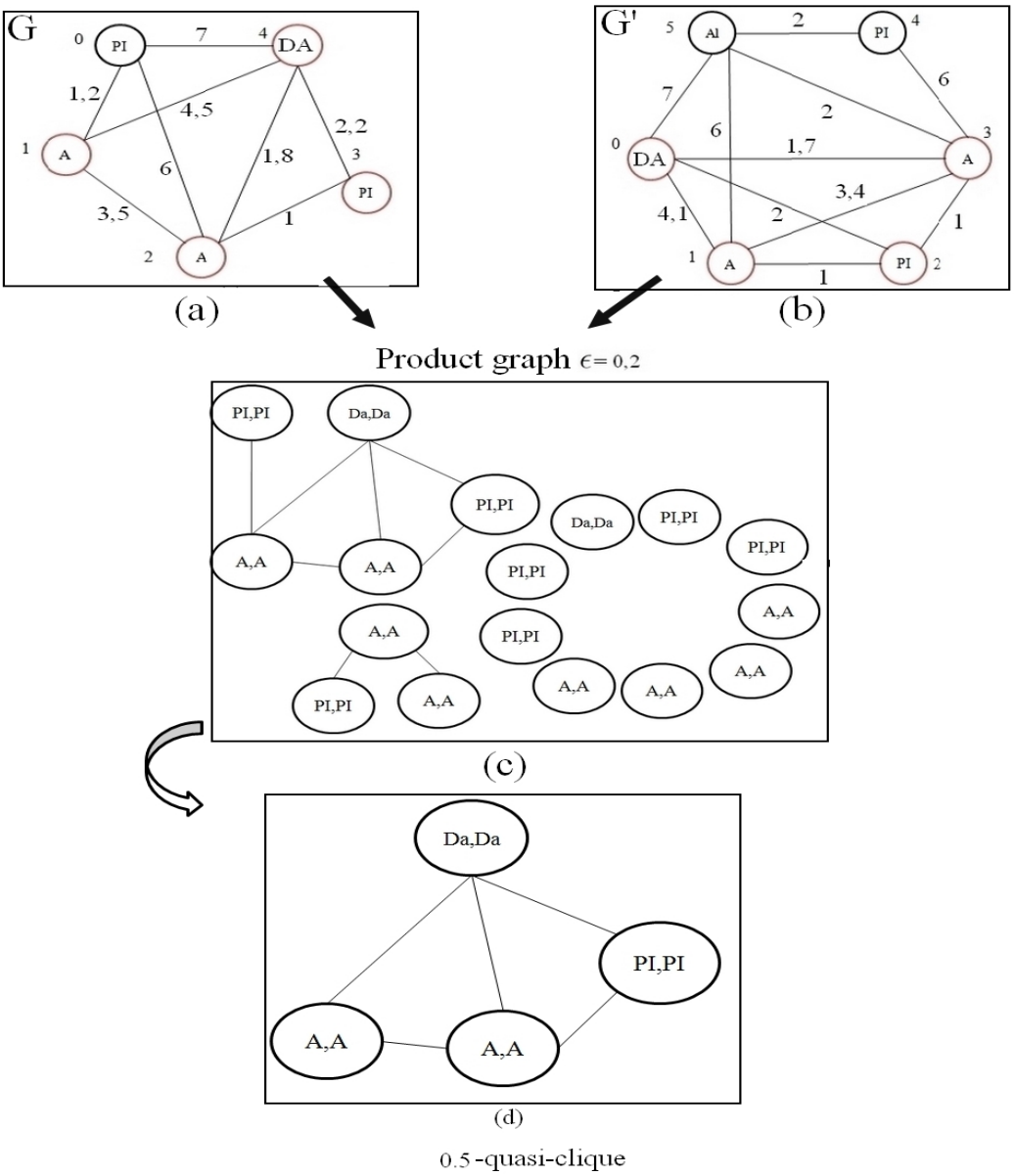

Figure 2.8: Maximum γ-quasi-clique detection

The objective is to extract a common substrucutre in order to measure the distance between the two graphs. To achieve this goal, an association graph G_p (c) is produced. This graph is composed of vertices sharing the same labels in both G and G', each vertex in G_p is represented by the two labels of the matched vertices. Also, the number near to each vertex is composed respectively of the number of the vertex in G (left) and G' (right). An edge is inserted in the product graph G_p if it exists in both graphs with a difference no more than the threshold ϵ which in this example is equal to 0.2*. Let $\gamma = 0.5$, whether we apply the merging-based method or the pruning-based method, the maximum 0.5-quasi-clique will be represented by (d) in Figure 2.8. Thus, the maximum 0.5-clique has a size of 4 nodes.*

2.4 Conclusion

In this chapter, we have presented the graph matching problem known from graph theory. In fact, we use graph matching in order to compare graphs representing structured objects. It consists of two subproblems: (i) Exact matching including graph and subgraph isomorphism which are very restrictive because they require that the two structures are identical and (ii) Inexact graph matching which introduces some error-tolerance. We described a method that provides some tolerance to exact matching, it consists of extracting an approximately common subgraph called γ-quasi-clique from a so called product graph which relaxes the *mcs*.

Then, we have extended the concepts developed in the previous chapter of protein functional annotation to a graph-based formalism in order to detect similarities between two protein binding sites structures. Unfortunately, the downward closure property does not hold for quasi-cliques thus, their mining on large graphs is a challenging task. The main idea behind this master thesis, is to replace the γ-quasi-clique by another type of a densely connected structure which is community. To this end, we make use of a heuristic approach, in which communities are merged iteratively into larger components in order to extract the maximum densest common subgraph as an alternative common substructure. In the following chapter, we will describe how communities detection can be used to measure similarities between two protein binding sites graphs and contributes to the problem of graph comparison.

Chapter 3

Graph-based similarity derived from community detection

3.1 Introduction

In many applications, it is necessary to compare objects represented as graphs and to determine the degree of similarity between them. This is accomplished using graph matching which is often formulated as a problem involving the maximum common subgraph isomorphism between the two graphs under consideration (Raymond & Willett, 2002). The *mcs* has been intensively studied in a number of domains such as chemistry, pattern recognition, computer vision and increasingly, biology for computing similarities between biomolecules. The similarity is measured using the size of the largest common substructure. The latter is an overly stringent measure characterized by a lack of flexibility and tolerance toward structural variation which is not convenient especially in the case of molecular data which is frequently noisy and incomplete (Liu & Limsoon, 2008). Furthermore, mining the *mcs* is an NP-complete problem, it is usually reduced to the extraction of the maximum clique from the product graph.

Yet, applying the clique concept is very stringent since it restrains the common substructure to the nodes linked by exactly the same number of edges in both graphs being considered. Besides, molecular data are frequently attained with noise and errors due to mutations and molecular flexibility. Consequently, it will penalize such distortions and produce a small common subgraph that is not able to capture structural similarities. Nevertheless, cliques may be replaced by the γ-clique concept which could provide some tolerance to the matching process (Fober et al., 2013). However, since the downward closure

does not hold for quasi-cliques, mining graphs with hundreds of nodes and edges is an intractable task.

To tackle these problems, we are interested in providing a more relaxed version of the maximum common subgraph measure (mcs) in order to increase its tolerance toward errors and noise with a low computational cost. We propose an alternative way based on the concept of community detection on graphs which has shown its efficiency in many real world applications (e.g., Balachander & Jia, 2000; Blondel et al., 2008; Fortunato, 2010).

This chapter describes our approach for assessing similarities between graphs representing protein binding sites, it is organized as follows: Section 3.2 presents the contribution of this work, it details the method used to detect communities and the way we use a community to assess similarities. Section 3.3, presents an illustrative example of the proposed approach.

3.2 Definitions and notations

Before presenting the approach of detecting communities, it is worth mentioning the difference between communities and the cliques and quasi-cliques concepts and present community detection basics that we will need to introduce the approach.

3.2.1 From cliques, quasi-cliques to communities

A clique is the densest form of graphs such that all nodes must be connected to each other. Indeed, it is a too strict condition. Similarly, a quasi-complete-subgraph is the relaxed version of the clique concept which requires for all nodes to be connected to at least $\gamma \cdot (|V| - 1)$ other ones. This is a more tolerant way than that of cliques but still has some limitations. To begin with, determining a maximum γ-clique in a graph is NP-complete. Furthermore, the choice of γ itself is perplexing. In fact, for small values of γ, the relaxation becomes overstated in such a way that geometrical constraints are ignored. For example, when $\gamma \to 0$, the subgraph will be composed of isolated vertices thus, the similarity will be based only on the physicochemical properties distribution without taking into account the binding sites geometry. On the other hand, for important values of γ, we return to the strictness problem. Hence, there is a trade-off problem between a small and large values of γ.

Nevertheless, a community refers to a sub-unit composed of a special group of vertices with high concentration of edges. It is a subgraph whose nodes are more tightly connected

to each other than with nodes outside the subgraph. A subgraph with all possible internal links except a few ones would not form a clique. Yet, it may be considered as a community. The most recent definitions of communities are based on the fact of comparing the internal and external cohesion of a subgraph links. In fact, as dense as it can be, it cannot be considered as a community if there is a strong connectedness as well between the subgraph and the rest of the graph (Fortunato, 2010). In other words, the notion of community is essentially based on the fact that edges inside the subgraph must be more than edges linking this subgraph vertices with the rest of the graph. A community may be a clique having all pairs of vertices linked, as well as a quasi-clique for internal edges concentration of order γ. In summary, a community may be considered as a generalization of the γ-clique concept and consequently the clique concept.

3.2.2 Basic concepts

3.2.2.1 Modularity

The modularity is one of the most popular quality functions given by (Newman & Girvan, 2004), it enables the evaluation of whether a particular division of a graph into communities is a good one which allows to quantify the success of the community finding algorithms. It can be applied for both weighted and unweighted graphs. It is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} [A_{i,j} - \frac{k_i k_j}{2m}] \delta(c_i, c_j) \tag{3.1}$$

Where $A_{i,j}$ is the adjacency matrix, k_i is the degree of vertex i, m is the total number of edges of the graph, $\delta(c_i, c_j)$ yields one if vertices i and j are in the same community (c_i, c_j), *zero* otherwise.

3.2.2.2 Modularity gain

ΔQ measures the variation of modularity resulting from moving an isolated vertex i into a neighboring community C, if i is already in a community, it is removed then added to C. It is defined as:

$$\Delta Q = [\frac{\sum_{in} + K_{i,in}}{2m} - (\frac{\sum_{tot} + K_i}{2m})^2 \ ] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{K_i}{2m})^2] \tag{3.2}$$

Where $\sum_{in}$ is the number of edges inside community C, $\sum_{tot}$ is the number of edges incident to community C, k_i is the degree of vertex i, $K_{i,in}$ is the number of edges incident from i to vertices in C and m is equal $|V|$.

3.2.2.3 Density

In order to measure the concentration of internal edges in a particular community (subgraph), we use a density function. The density δ_C (Li et al., 2005) of a community $C(V_C, E_C)$ is given by the ratio between the number of its internal links and the number of all possible internal links, it is defined as follows:

$$\delta_C = \frac{2 \times |E_C|}{|V_C| \times |V_C - 1|} \tag{3.3}$$

Obviously, $0 \leqslant \delta_C \leqslant 1$, it is equal 1 when C is fully connected (clique). For C to be a community in $G(V, E)$, it would have a larger density δ_C than the average density of G which is given by the ratio between the number of edges $|E|$ and the maximum number of possible edges $\frac{|V| \times (|V|-1)}{2}$.

3.3 The community detection based approach

Community detection in graphs has gained much attention in the previous decades in numerous areas where systems are modeled as graph networks such as biology, sociology, economy and computer science, to name but a few. The main idea is to partition the graph into cohesive subunits (Fortunato, 2010). In fact, dense regions may indicate interesting properties depending on what characteristics are being modeled by the graph vertices and edges. They may reveal mutual similarity, interaction degrees or favorable environments, etc (Charalampos et al., 2013). We use the principle of community detection to extract a dense substructure in order to measure the similarity between two graphs. Following is the proceedings of our approach.

3.3.1 Graph-based similarity using community detection

In this subsection, we present the details of the different steps to assess the similarity between a pair of graphs. The structure of ââour approach is as follows:

- In a first step, given two graphs $G(V, E)$ and $G'(V', E')$ representing two protein binding sites that we aim to compare, we produce their association graph in order to conserve the graphs structures (structure of nodes, edges and labels).

- In a second step, we partition the product graph into dense subunits without restrictions to the degree of connectedness of vertices (unlike the clique and quasi-clique detection methods). Therefore we apply a method of detecting communities inspired of the algorithm proposed in (Blondel et al., 2008) which is one of the best algorithms proposed in the context of communities detection. It showed its effectiveness in partitioning a Belgian mobile phone network. It is a greedy approach based on the optimization of the so called modularity function of Newman and Girvan (Newman & Girvan, 2004) (Equation 3.1). We have made ââtechnical changes to this algorithm to make it suitable to our context of almost maximum common subgraph detection.

  The product graph partitioning is based on a sequence of passes repeated iteratively. At the beginning, each node is considered as a community apart. During each pass, nodes are merged into their neighborhood and evaluated by the modularity variation illustrated in Equation 3.2. A node is placed in the community for which the modularity gain is positive and maximum. A pass ends when all nodes have been considered. The algorithm stops when no further improvement can be accomplished and hence a local maxima of the modularity (Equation 3.1) is attained. As a result, the original graph will be divided into sub-units that are highly connected.

  Note that a community cannot contain two nodes that correspond to the same node in one of the considered graphs in order to ensure the validity of common subgraphs approximation. In fact, product graph nodes refer to unique pairs $(v_i, v'_j) \in V \times V'$. Yet, many product nodes may refer to the same nodes in G or G'. Therefore, before adding a new product node into a community we check whether the nodes it refers to are not yet contained.

- Next is the almost maximum common subgraph detection. First, we eliminate communities representing cliques since the whole objective is to relax the clique condition. Then, we use a minimum size treshold (min_size) in order to filter small communities. Finally we extract the maximum densest common community ($MDCC$) that maximizes δ_C from Equation 3.3 with maximum number of nodes. In sum, it is a more flexible way to extract an approximation of the largest common subgraph between the two original graphs that is rather dense.

- The last step involves the similarity computation which will be presented using the following measure.

In short, we can summarize the steps of our approach into the five following steps:

1. Product graph construction: Build it from two graphs representing protein binding sites.
2. Communities detection: Partition the product graph into dense sub-units.
3. Cliques elimination: Eliminate communities whose density is equal 1 (cliques).
4. Maximum densest common community $MDCC$ extraction: Extract the maximum densest community whose size > min_size.
5. Similarity computation: Consider the extracted $MDCC$ as a maximum "almost" common subgraph and use it to compute the similarity between the two graphs as explained in the following subsection.

3.3.2 Similarity measure based on the densest maximum common community

As mentioned in Chapter 2, more precisely in Section 2.2, the *mcs* produces a numerical degree of similarity using Equation 2.1. In fact, the size of the maximum detected community can in turn be used to define a similarity degree using this metric. However, it was shown that in some cases on protein binding sites graphs, a structure would be larger than the substantial binding site. The underlying reason for that is that protein binding pockets do not have a clear cut boundary. Thus, a structure can be a subpocket of another one having the most important catalytic residues. From this point of view, (Boukhris et al., 2009) expressed the similarity between two graphs in terms of subset relations, given two graphs $G = (V, E)$ and $G' = (V', E')$ and their maximum common subgraph $G_{max}(V_{max}, E_{max})$, they considered $G \subseteq G'$ and $G' \subseteq G$ instead of $G = G'$ which was inspired from set theory. They made use of the Ordered Weighted Averaging (OWA) operator proposed in (Yager, 1988) which provides a generalization to using the minimum operator for combining the two degrees of inclusion leading to the following measure:

$$sim(G, G') = \varphi min(\alpha, \beta) + (1 - \varphi) max(\alpha, \beta) \tag{3.4}$$

Where $0 < \varphi < 1$, is a compromise parameter which determines the comparison type (equivalence/inclusion). $\alpha = \frac{|V_{max}|}{|V|}$ expresses the degree to which G is a subset of G' and $\beta = \frac{|V_{max}|}{|V'|}$ is the degree to which G' is a subset of G. We can apply this similarity measure

using our maximum detected community $C(V_C, E_C)$. Thus, $\alpha = \frac{|V_C|}{|V|}$ and $\beta = \frac{|V_C|}{|V'|}$. In general, the larger the value of $sim(G, G')$, the more similar G is to G'. This measure is more suitable, especially in the context of evaluating similarities between protein binding sites.

3.3.3 Complexity

The heuristic applied here is a modularity optimization based-method. In fact, an exhaustive optimization of the modularity is impossible because there are various ways in which it is possible to partition a graph. Besides, it has been proved recently that modularity optimization is an NP-complete problem. Thus, the true maximum is out of reach. However, there are several algorithms able to find good approximations of the modularity maximum in a reasonable time, one of which is the algorithm we have inspired our method from. It is intuitive and easy leading to an unsupervised outcome. The algorithm is very fast especially for sparse graphs (Blondel et al., 2008).

In fact, upon each iteration, one has to compute the modularity variation ΔQ, this computation can be done at constant time, it requires a time $O(m)$ where m is the number of edges. Besides, the modularity update after communities merging during each iteration can be done in worst case $O(n)$ where n is the number of communities which decreases quickly after just few iterations so that the most of the running time is concentrated on the first ones (Fortunato, 2010). Although the theoretical complexity of the algorithm is not discussed, at the experimental level, the method is evaluated to run in time $O(nlog(n))$ which makes it the fastest algorithm for identifying communities (Pérez et al., 2014). The technique is more limited by storage than by computational time (Blondel et al., 2008).

3.3.4 Our contributions

In this work, we contribute to the problem of assessing similarities between node-labeled and edge-weighted graphs representing protien binding sites. We relax the problem of exact matches based on the maximum common subgraph isomorphism using a new concept that is rather dense that we call the maximum densest common community $MDCC$. Thus, we provide flexibility against noise and perturbations and increase robustness toward structural variation and ensure a reliable prediction of a protein class by searching for similarities between binding sites structures independent of sequence or fold information. The $MDCC$ detection is characterized by a low complexity thus, it is expected to be detected in a low cost computational time.

3.4 Illustration example

Let us illustrate the community detection method by a simple example. Let be two graphs representing two binding sites denoted: $G(V, E)$ and $G'(V', E')$ as shown in Figure 3.1. In order to extract a common substructure from these two graphs, we first derive a product graph ($\epsilon = 0.2$) as explained in Chapter 2, more precisely in Section 2.2.

3.4.1 The product graph construction step

The steps of the product graph $G_p(V_p, E_p)$ construction are enumerated as follows:

1. For each $v_i \in V$ and $v_j \in V'$, we compare v_i and v_j labels and we insert a product node $V_{p_i}(v_i, v_j)$ is they are equal.
2. For each $(v_i, v'_i) \in V_p$ and $(v_j, v'_j) \in V_p$, we compare the edges weights' separating v_i, v'_i and v_j, v'_j and insert an edge between (v_i, v'_i) and (v_j, v'_j) if their weights are less or equal the threshold ϵ.
3. For each isolated node $V_{p_i} \in V_p$, we prune V_{p_i}. This pruning step is performed in order to reduce the false positive matches and non trivial substructures that are not significant in a biological sense.

3.4.2 The communities detection step

We apply the community detection algorithm. The algorithm is divided into a number of iterations that are repeated until a local maxima of the modularity Q (Equation 3.1) is attained. In the following, we present the progression of this communities detection phase:

- At the beginning, each node is assigned to a different community. Thus, we have as many communities as we have vertices. As illustrated, the number of edges m in the product graph is equal 9.
- For each vertex i, we consider its neighbors and we evaluate the modularity variation ΔQ (Equation 3.2) obtained when adding i in its adjacent community. i is assigned to the community for which the gain is maximum and positive and especially if this community does not contain a vertex j such that i and j refer to the same node in G or G'. Otherwise, the considered vertex remains in its community. This

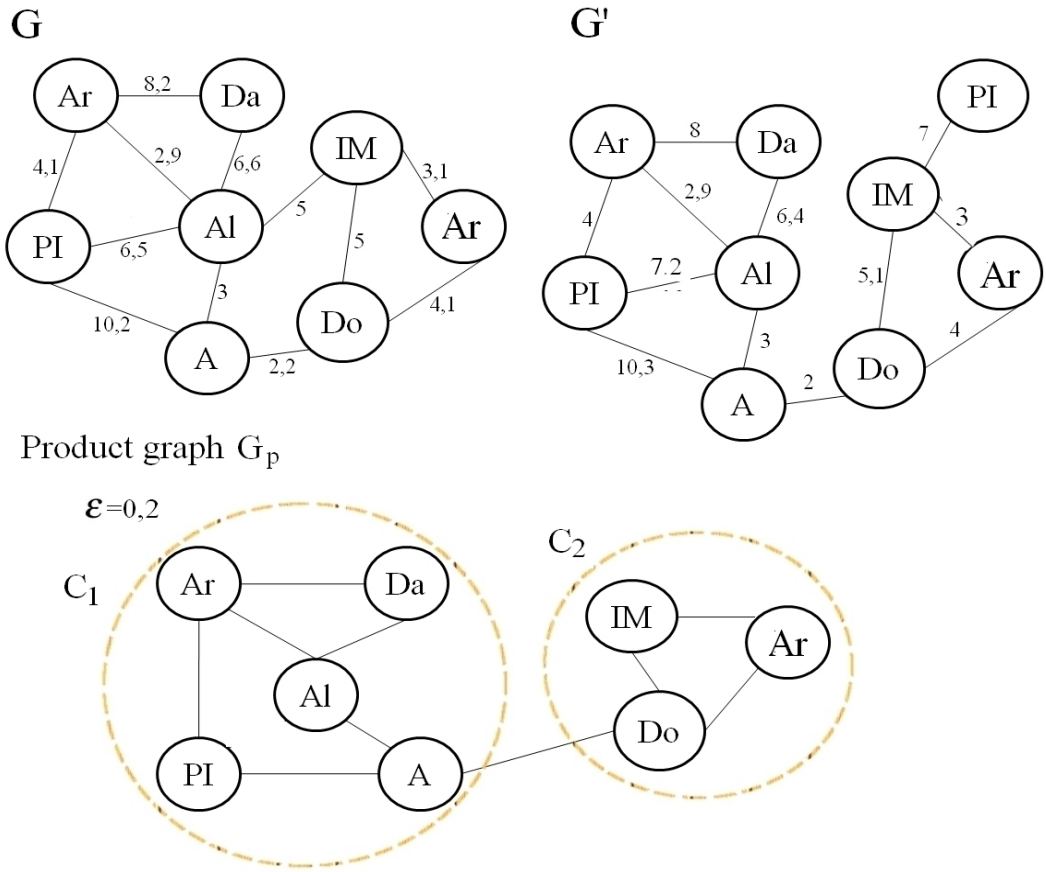

Figure 3.1: Illustration of the community detection algorithm operating on a protein binding site product graph

process is repeated until no further improvement can be achieved. In this example, we do not have nodes that refer to the same node in one of the two graphs thus, the testing of same graph step is not illustrated. It is worth mentioning that the order in which the vertices are considered does not have a significant influence on the algorithm output (Blondel et al., 2008). Suppose we generate a random order as follows: $Ar, Da, A, Al, PI, Do, IM, Ar$. A vertex may be considered several times.

Given the product graph G_p shown in Figure 3.1, the iterations of the algorithm will be as follows:

First pass:

- $Q = 0$
- We consider the first vertex $i = Ar$

- $\Delta Q((Ar, Da), Ar) = [\frac{1+1}{18} - \frac{3+3}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{3}{18})^2] = 0$
- $\Delta Q((Ar, Al), Ar) = [\frac{1+1}{18} - \frac{4+3}{18}^2] - [\frac{1}{18} - (\frac{4}{18})^2 - (\frac{3}{18})^2] = -0.018$
- $\Delta Q((Ar, PI), Ar) = [\frac{1+1}{18} - \frac{3+3}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{3}{18})^2] = 0$

$\longrightarrow$ no gain of modularity, Ar remains in its community.

- $i = Da$

- $\Delta Q((Da, Ar), Da) = [\frac{1+1}{18} - \frac{3+2}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{2}{18})^2] = 0.018$
- $\Delta Q((Da, Al), Da) = [\frac{1+1}{18} - \frac{3+2}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{2}{18})^2] = 0.018$

$\longrightarrow$ we choose randomly between Ar and Al, Da merges to the community (Ar, Da)

- $Q = \frac{1}{18}[1 - \frac{2\times3}{18}] = 0.037$
- $i = Al$

-$\Delta Q((Al, (Ar, Da)), Al) = [\frac{3+2}{18} - \frac{2+3}{18}^2] - [\frac{3}{18} - (\frac{2}{18})^2 - (\frac{3}{18})^2] = 0.074$

- $\Delta Q((Al, A), Al) = [\frac{1+1}{18} - \frac{4+2}{18}^2] - [\frac{1}{18} - (\frac{4}{18})^2 - (\frac{3}{18})^2] = -0.018$

$\longrightarrow$ Al merges to community (Ar, Da, Al)

- $Q = \frac{1}{18}[1 - \frac{2\times3}{18} + 1 - \frac{3\times3}{18} + 1 - \frac{2\times3}{18}] = 0.101$
- $i = A$

- $\Delta Q((A, C_1), A) = [\frac{4+1}{18} - \frac{3+3}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{3}{18})^2] = 0$
- $\Delta Q((A, PI), A) = [\frac{1+1}{18} - \frac{3+3}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{3}{18})^2] = 0$

$\longrightarrow$ no gain of modularity, A remains in its community

- $Q = 0.101$
- $i = PI$

- $\Delta Q((PI, C_1), PI) = [\frac{4+1}{18} - \frac{2+2}{18}^2] - [\frac{1}{18} - (\frac{2}{18})^2 - (\frac{2}{18})^2] = 0.030$
- $\Delta Q((PI, A), PI) = [\frac{1+1}{18} - \frac{3+2}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{2}{18})^2] = 0.018$

$\longrightarrow$ PI merges to community (Ar, Da, Al, PI),

- $Q = \frac{1}{18}[1 - \frac{2\times3}{18} + 1 - \frac{3\times3}{18} + 1 - \frac{2\times3}{18} + 1 - \frac{3\times2}{18}] = 0.138$
- $i = Do$

  - $\Delta Q((Do, A), Do) = [\frac{1+1}{18} - \frac{4+3}{18}^2] - [\frac{1}{18} - (\frac{4}{18})^2 - (\frac{3}{18})^2] = -0.018$
  - $\Delta Q((Do, Ar), Do) = [\frac{1+1}{18} - \frac{2+3}{18}^2] - [\frac{1}{18} - (\frac{2}{18})^2 - (\frac{3}{18})^2] = 0.018$
  - $\Delta Q((Do, IM), Do) = [\frac{1+1}{18} - \frac{2+3}{18}^2] - [\frac{1}{18} - (\frac{2}{18})^2 - (\frac{3}{18})^2] = 0.018$

  $\longrightarrow$ we choose randomly between Ar and IM, Do merges to community (IM, Do)

- $Q = \frac{1}{18}[1 - \frac{2\times3}{18} + 1 - \frac{3\times3}{18} + 1 - \frac{2\times3}{18} + 1 - \frac{3\times2}{18} + 1 - \frac{3\times2}{18}] = 0.175$
- $i = IM$

  - $\Delta Q((IM, Do), IM) = [\frac{1+1}{18} - \frac{3+2}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{2}{18})^2] = 0.018$
  - $\Delta Q((IM, Ar), IM) = [\frac{1+1}{18} - \frac{2+2}{18}^2] - [\frac{1}{18} - (\frac{2}{18})^2 - (\frac{2}{18})^2] = 0.030$

  $\longrightarrow$ IM merges to community (Im, Ar)

- $Q = \frac{1}{18}[1 - \frac{2\times3}{18} + 1 - \frac{3\times3}{18} + 1 - \frac{2\times3}{18} + 1 - \frac{2\times2}{18}] = 0.182$
- $i = Ar$

  - $\Delta Q((Ar, Do), Ar) = [\frac{1+1}{18} - \frac{3+2}{18}^2] - [\frac{1}{18} - (\frac{3}{18})^2 - (\frac{2}{18})^2] = 0.018$
  - $\Delta Q((Ar, IM), Ar) = [\frac{1+1}{18} - \frac{2+2}{18}^2] - [\frac{1}{18} - (\frac{2}{18})^2 - (\frac{2}{18})^2] = 0.030$

  $\longrightarrow$ Ar remains in its current community (Im, Ar), $Q = 0.182$.

- Communities are:$\{(Ar, Da, Al, PI), (A), (IM, Ar), (Do)\}$

Second pass:

- $i = Ar$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = Da$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation

- $i = A$

  - $\Delta Q((Ar, Da, Al, PI, A), A) = [\frac{6+2}{18} - \frac{1+3}{18}^2] - [\frac{6}{18} - (\frac{1}{18})^2 - (\frac{3}{18})^2] = 0.0925$
  - $\Delta Q((Do, A), A) = [\frac{1+1}{18} - \frac{4+3}{18}^2] - [\frac{1}{18} - (\frac{4}{18})^2 - (\frac{3}{18})^2] = -0.018$

  $\longrightarrow$ A merges to community (Ar, Da, Al, PI, A)

- $Q = \frac{1}{18}[1 - \frac{2\times3}{18} + 1 - \frac{3\times3}{18} + 1 - \frac{2\times3}{18} + 1 - \frac{3\times2}{18} + 1 - \frac{3\times3}{18} + 1 - \frac{3\times2}{18}] = 0.246$
- $i = Al$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = PI$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = Do$:

  - $\Delta Q((Do, Ar, Da, Al, PI, A), Do) = [\frac{7+1}{18} - \frac{2+3}{18}^2] - [\frac{7}{18} - (\frac{2}{18})^2 - (\frac{3}{18})^2] = 0.018$
  - $\Delta Q((Do, IM, Ar), Do) = [\frac{3+2}{18} - \frac{1+3}{18}^2] - [\frac{3}{18} - (\frac{1}{18})^2 - (\frac{3}{18})^2] = 0.0925$

  $\longrightarrow$ Do merges to community (Do, IM, Ar)

- $Q = \frac{1}{18}[1 - \frac{2\times3}{18} + 1 - \frac{3\times3}{18} + 1 - \frac{2\times3}{18} + 1 - \frac{3\times2}{18} + 1 - \frac{3\times3}{18} + 1 - \frac{3\times2}{18} + 1 - \frac{2\times3}{18} + 1 - \frac{2\times3}{18}] = 0.320$
- $i = IM$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = Ar$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- Communities are:$\{(Ar, Da, Al, PI, A), (IM, Ar, Do)\}$

Third pass:

- $i = Ar$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = Da$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = A$:

- $\Delta Q((Do, Ar, Da, Al, PI, A), A) = [\frac{6+2}{18} - \frac{1+3}{18}^2] - [\frac{6}{18} - (\frac{1}{18})^2 - (\frac{3}{18})^2] = 0.0925$
- $\Delta Q((A, Do, IM, Ar), A) = [\frac{4+1}{18} - \frac{2+3}{18}^2] - [\frac{4}{18} - (\frac{2}{18})^2 - (\frac{3}{18})^2] = 0.018$

$\longrightarrow$ A stays in community (Do, Ar, Da, Al, PI, A) $\longrightarrow$ No modularity variation

- $i = Al$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = PI$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = Do$:
  - $\Delta Q((Do, Ar, Da, Al, PI, A), Do) = [\frac{7+1}{18} - \frac{2+3}{18}^2] - [\frac{7}{18} - (\frac{2}{18})^2 - (\frac{3}{18})^2] = 0.018$
  - $\Delta Q((Do, IM, Ar), Do) = [\frac{6+2}{18} - \frac{1+3}{18}^2] - [\frac{6}{18} - (\frac{1}{18})^2 - (\frac{3}{18})^2] = 0.0925$

  $\longrightarrow$ Do stays in community (Do, IM, Ar) $\longrightarrow$ No modularity variation
- $i = IM$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- $i = Ar$: All of its neighboring belongs to the same community $\longrightarrow$ No modularity variation
- Communities are:$\{(Ar, Da, Al, PI, A), (IM, Ar, Do)\}$
- No modularity improvement $\longrightarrow$ Final $Q = 0.320$

Finally, we obtain two communities $C_1 = (Ar, Da, Al, PI, A)$ and $C_2 = (IM, Ar, Do)$ as shown in Figure 3.1.

3.4.3 The cliques elimination step

After applying the community detection algorithm we obtain two communities C_1 (5 nodes) and C_2 (3 nodes), denoted by the dashed circles in Figure 3.1.

The next step is the elimination of communities having density equals 1 (cliques). Therefore, we remove community C_2 represented in Figure 3.1.

3.4.4 The maximum densest common community $MDCC$ extraction step

We choose the densest community having at least four vertices (suppose $min_size = 4$). Since no more than one community is left, we choose community C_1. Thus, the largest community, representing the maximum almost common subgraph, has a size of 5 nodes and a density $\delta_C = \frac{2\times 6}{5\times(5-1)} = 0.6$. If we consider the clique detection algorithm, there would be subgraphs having up to 3 vertices. Therefore, the maximum clique corresponding to the maximum common subgraph will have a size of 3 nodes as illustrated in Figure 3.2 (a) which describes the possibles maximum cliques. (Figure 3.2 (b))

Besides, considering the γ-clique concept, depending of the maximum quasi-clique detection method we will have different results. Indeed, if we consider the merging based method (Fober et al., 2013), for any value of $\gamma < 0.6$, the maximum quasi-clique will have a size of 5 nodes as illustrated in (Figure 3.2 (b)). For $\gamma > 0.6$, no quasi-clique will be outputted.

On the other hand, if we apply the pruning-based method (Boukhris et al., 2009) and take $\gamma \in [0.1, 0.2]$ or 0.3 as illustrated in (Figure 3.2 (c)). The maximum γ-clique will be composed respectively of 8 and 7 vertices and hence will represent the whole (almost whole for $\gamma = 0.3$) product graph which is not meaningful because geometrical constraints are mainly ignored. If we take $\gamma \in [0.4, 0.5]$, the size of the maximum γ-clique will be composed of 5 nodes (Figure 3.2 (c)). If we set $\gamma \geqslant 0.6$ (Figure 3.2 (c)), the maximum γ-clique will be composed of less than 5 vertices. Thus, for $\gamma \in [0.4, 0.5]$ we obtain a substructure of the same size of that obtained when applying our community detection based method. However, not only we have struggled to choose the appropriate γ value but also it is expensive in terms of computation time to detect the maximum γ-clique.

Accordingly, the community detection method allows the detection of a rather large common substructure which have dense internal links and that remains biologically significant.

3.4.5 The similarity computation step

The final step consists at measuring the similarity between the two considered graphs. We use Equation 3.4, let $\varphi = 0.5$ and $\gamma = 0.5$. The similarity between the two graphs G and G' for the different methods will be:

- Maximum clique: $sim(G, G') = 0.5 * min(\frac{3}{8}, \frac{3}{9}) + (1 - 0.5)max(\frac{3}{8}, \frac{3}{9}) = 0.35$.

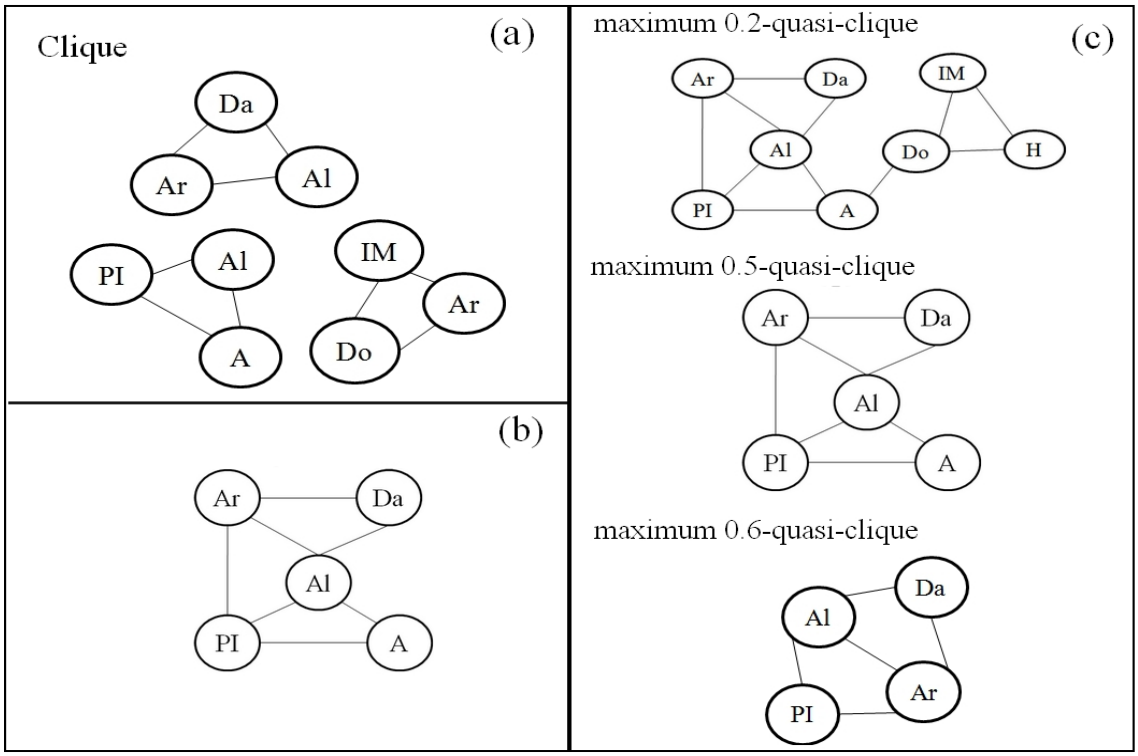

Figure 3.2: Maximum clique (a) and quasi-clique (b) detection

- Maximum γ-clique[1]:$sim(G, G') = 0.5 * min(\frac{5}{8}, \frac{5}{9}) + (1 - 0.5)max(\frac{5}{8}, \frac{5}{9}) = 0.59$.
- Maximum γ-clique[2]:$sim(G, G') = 0.5 * min(\frac{5}{8}, \frac{5}{9}) + (1 - 0.5)max(\frac{5}{8}, \frac{5}{9}) = 0.59$.
- Maximum densest community: $sim(G, G') = 0.5 * min(\frac{5}{8}, \frac{5}{9}) + (1 - 0.5)max(\frac{5}{8}, \frac{5}{9}) = 0.59$.

3.5 Conclusion

In this chapter, we presented our approach for the assessment of the similarity between graphs representing protein binding sites. Our method is a graph-based similarity derived from the detection of the maximum almost common subgraph which is community. It is a relaxation of the *mcs* measure which allows to have larger common substructures between the input graphs. We have presented the method used to detect the maximum community. Then, we have detailed the different steps allowing to compute the degree of similarity

[1]Merging-based method
[2]Pruning-based method

between two graphs.

The next chapter is devoted to the experimental study. We will present the major algorithms used in order to demonstrate the effectiveness of our method. We will compare it the standard clique measure and the γ-clique measure. Then, we will report different results obtained from simulations performed on real databases.

Chapter 4

Experimental study

4.1 Introduction

In order to evaluate our method, we conducted an experimental study in the context of comparing protein binding sites in order to predict a protein unknown family (i.e., class). This is important because it allows us to have an idea about the feasibility of our new method. In this chapter, we report the programs developed for the implementation of our approach, they are developed in Java language with Netbeans 7.4.

Besides, we have performed simulations on real datasets obtained from the CavBase database (Schmitt et al., 2002) in order to check the effectiveness of our method. The results will be reported and analyzed.

This chapter is organized as follows. First, Section 4.2 details the implementation of our graph similarity measure derived from communities detection. It exposes the principal variables, programs and algorithms. Section 4.3 is devoted to the experimentation phase, it reports the results obtained over data representing protein binding sites.

4.2 Implementation

The implementation of our approach is done using the java language. Details of the developed programs are described in the following subsections.

4.2.1 Framework

The developed programs have as inputs:

- A dataset in the ".rlbcoor" format from CavBase (Schmitt et al., 2002), a database system for the fully automated detection and extraction of putative protein binding sites derived from crystal structure data retrieved from the PDB. In CavBase, the description of the binding sites is done using graphs structures.
- The cutoff value for edges length distances.
- The value of the tolerance threshold for edges weights differences.
- The minimum size required to allow the detection of the densest community.
- The neighbors' number for the K-NN classifier.

The outputs of our programs are:

- The maximum densest community on the product graph for two graphs.
- The similarity between the two graphs.
- The accuracy of the similarity measure given by the Percent of Correct Classification (PCC) (a popular criterion for measuring the accuracy of classifiers). It will be detailed in the following sections.

4.2.2 Principle variables

The main variables of our programs are the following:

- *path_file:* The path for the graphs input file.
- *min_size:* a minimum size threshold for filtering small communities.
- φ*:* the similarity measure weight criteria, its value is in [0,1].
- ϵ*:* the tolerance threshold value for edge length differences (0.2).
- δ*:* the upper bound value for edge distances (11Â).

- *k:* the number of closest neighbors that should be taken into account for the K-NN classifier.
- *Instances:* a vector of strings containing all the names of the files of the dataset.
- *Struct:* a string that indicates the name of a given structure.
- *graph:* an object that is composed of:
  - *.path:* a string that indicates the path of the file of the graph.
  - *.name:* the name of the corresponding molecule.
  - *.numberofvertices:* the number of vertices (physicochemical properties) of the graph.
  - *.vertexlabel:* a list of string containing the labels of the vertices.
  - *.distance:* a matrix of doubles containing the distances (weights) between the vertices.
  - *.vertices:* a list of integers containing the vertices indices.
- *product_graph_node:* a structure that incorporates a vector of two nodes: $node_i$ from a first graph G_i and a $node_j$ from a second graph G_j.
- *product_graph:* an object composed of:
  - *.edges:* a boolean matrix that represents the adjacency matrix.
  - *.nodes:* a list of product graph nodes.
  - *.productnodes:* a vector of product graph node.
  - *.size:* number of nodes of the product graph.
- *neighbors:* a list containing the set of the neighbors of a particular product graph node.
- *community:* a subgraph of the product graph composed of:
  - *.edges:* a boolean matrix that represents the adjacency matrix.
  - *.nodes:* a list of nodes.
  - *.numberOfnodes:* number of nodes.
- *neigh_comm:* a list containing the set of neighboring communities of a node.
- *comm_tot:* the number of edges incident to a community.
- *comm_in:* the number of edges inside a community.

- *k_i_in:* the number of edges incident from a node i to nodes in a particular community.
- *dens_comm:* a double variable that quantifies the density of a community.
- *class:* the functional class of the protein binding site.
- *sim:* a numerical degree representing the similarity between two graphs.
- *Similarities:* a matrix containing all pairwises comparisons.
- *PCC:* the percent of correctly classified protein binding sites classes.

4.2.3 Main programs

This subsection is dedicated to the major programs that we have implemented to build our method. They are regrouped according to the task they perform.

✓ ***Graph creation:***

- *Parse:* creates the list of nodes labels and the graph distance matrix and eliminates edges with a distance exceeding the cutoff value. It takes as parameter the path of the dataset file containing the binding sites in ".rlbcoor" format to parse.
- *Create_graph:* calling the Parse procedure, it creates a new graph.

✓ ***Product Graph construction***

- *Product_graph_node:* creates a product graph node resulting from the merging of two nodes sharing the same label in the input graphs.
- *Product_graph:* constructs a product graph from a pair of graphs.
- *Prune_Vertex:* prunes vertices in the product graph if they are not linked to any other ones.

✓ ***Communities detection***

Find_Communities: A principal iterative function that partition a graph according to the modularity improvement and output the set of communities. It uses these following functions:

- *Mod_Gain:* evaluates the gain of merging a node into a neighboring community.
- *Test_same_comm:* returns 1 if two nodes are in the same community.
- *Tot:* computes the total number of edges incident to a community.

- *In:* computes the total number of edges inside a community.
- *K_i_in:* computes the number of edges incident from a node to nodes inside a community.
- *Modularity:* returns the goodness of the current partition.
- *Density:* computes the density of a community.
- *Test_same_graph:* a boolean function that takes two nodes as input and returns false if these two do not refer to the same node neither in G_1 nor G_2, true otherwise.

✓ ***Measuring similarity***

- *Similarity:* evaluates the similarity between a pair of graphs using Equation 3.4 and save it into the matrix *Similarities*.
- *Eval_Sim:* evaluates our method by computing its PCC: the percentage of correctly classified graphs according to the family class they belong to.

4.2.4 Graph-based similarity derived from the communities detection algorithms

In this section, we present the major algorithms relative to our approach to measure the similarity between two graphs structures corresponding, from the graph creation to the similarity measurement.

- **Graph creation:**

1: **Input:** path_file, δ
2: **Output:** Graph G
3: **Begin**
4: Parse(path_file)
5: */* Eliminating edges with distances exceeding the upper bound value */*
6: trimEdges(δ)
7: */* Creating a list of nodes labels */*
8: CreateNodes()
9: */* Creating a graph G from a given binding site file */*
10: G $\leftarrow$ Graph(size, distances, nodes, coordinates, path)
11: **End**

- **Product graph construction:**

```
Input: G(V,E),G'(V',E'),ϵ
Output: Product_graph
Begin
/* Creating product graph nodes */
for all v_i ∈ G.vertices do
    for all v_j ∈ G'.vertices do
        if G.vertexlabel(v_i) = G'.vertexlabel(v_j) then
            product_graph_node.node1← v_i
            product_graph_node.node2← v_j
            productnodes.Add(product_graph_node)
        end if
    end for
end for
for all v = (v_i, v_j) ∈ productnodes do
    for all v' = (v'_i, v'_j) ∈ productnodes do
        /* Connecting product graph nodes */
        if Math.abs(G.distances[v_i][v_j])-(G'.distances[v'_i][v'_j])≤ ϵ then
            /* Set the corresponding cell of the adjacency matrix to True*/
            edges[i][j]← true
            edges[j][i]← true
        else if
             then/* Set it to False otherwise */
            edges[i][j]← true
            edges[j][i]← true
        end if
    end for
end for
/* Prune product graph isolated nodes */
for all node_i ∈ productnodes do
    if neighbors(i)=null then
        Prune_Vertex(i)
    end if
end for
End
```

- **Communities detection**

```
1: Input: Product_graph
2: Output: Communities /* A set of communities */
3: Begin
4: for all v_i ∈ Product_graph.nodes do
5:     /* Assign a new community to each node */
6:     CreateNewCommunity(C_i, node_i)
7: end for
8: /* Repeat nodes merging into communities while there is a modularity improvement */
9: while Improvement=True do
10:     /* Store the current value of the modularity */
11:     Curr_Mod ← Modularity()
12:     for all v_i ∈ Product_graph.nodes do
13:         /* Compute the modularity gain if node v_i stays in C_i */
14:         ΔQ(i, C_i) ← ΔQ(Tot(C_i), In(C_i), K_i_in(v_i, C_i))
15:         C_i.Remove(v_i)
16:         for all C_j ∈ neigh_comm(v_i) do
17:             /* Check if v_i and one of C_j nodes refer to the same node */
18:             if Test_same_graph(v_i, C_j) = false then
19:                 /* Insert v_i into its neighbor community C_j */
20:                 Community(v_i) ← C_j
21:                 /* Compute the modularity gain of merging node v_i into C_j */
22:                 ΔQ(i, C_j) ← Δ_Q(Tot(C_j), In(C_j), K_i_in(v_i, C_j))
23:                 /* Store the maximum gain and the optimal community */
24:                 if Δ_Q > 0 and Δ_Q > Δ_Qmax then
25:                     Δ_Qmax ← Δ_Q
26:                     C_opt ← C_j
27:                 end if
28:             end if
29:         end for
30:         /* Assign v_i to the community that maximizes Δ_Q */
31:         if Δ_Qmax > ΔQ(i, C_i) then
32:             Community(v_i) ← C_opt
33:             New_Mod ← Modularity()
34:         else
35:             Community(v_i) ← C_i
36:             New_Mod ← Modularity()
37:         end if
```

```
        /* Check if there is a modularity improvement */
        if New_Mod=Curr_Mod then
            Improvement=False
        end if
    end for
end while
End
```

- $MDCC$_**det**

```
Input: G,G',min_size,ϵ
Output: C_max /* maximum densest common community */
Begin
/* Construct the product graph of G and G' */
ProdGraph ← Product_Graph(G,G',ϵ)
Communities ← Find_communities(ProdGraph)
/* Eliminate communities that form a clique */
for all C_i ∈ Communities do
    if C_i.density=1 then
        Communities.Remove(C_i)
    end if
end for
for all C_i ∈ Communities do
    /* Get the densest community that satisfies a minimum number of nodes */
    if C_i.density> D_max and C_i.numberOfnodes ≥ min_size then
        C_max ← C_i
        D_max ← C_i.density
    end if
end for
End
```

- **Similarity assessment**

```
Input: δ,query_path_file,φ,sample_path_file,ϵ
Output: Simil
Begin
G ← Parse(query_path_file, δ)
G' ← Parse(sample_path_file, δ)
/* Computing the size of the maximum community found between G and G' */
```

```
C_max ← MDCC_det(G, G', min_size,ϵ)
Size_max ← C_max.numberOfnodes
α ← Size_max/G.size()
β ← Size_max/G'.size()
if α > β then
    max ← α
    min ← β
else
    max ← β
    min ← α
end if
Similarity[G][G'] ← φ × min + (1 − φ) × max
End
```

- **Eval_Sim**

```
Input: Merge,File,δ, φ, ϵ
Output: PCC
Begin
/* Making all pairwises comparisons */
Similarities ← classif(Instances, δ,ϵ,φ,min_size)
/* Sorting Similarities */
Similarities ← sort(Similarities)
/* For each row in Similarities find the K most similar neighbors */
Similarities ← Kmatrix(Similarities)
for all i ∈ Similarities do
    /* For each structure (row i) predict its class according to the vote of the K-NN classifier */
    class ← Vote(Similarities[i])
    if Verify(true_class, class)=true then
        Correct ← Correct+1
    end if
end for
/* Computing PCC */
PCC ← (Correct.size/Instances.size) ×100
End
```

4.3 Experimentations

In order to evaluate our new method. We have performed tests on protein binding sites real datasets obtained from the CavBase database. The results of these simulations will be presented and analyzed in the following subsections.

4.3.1 Evaluation criteria

4.3.1.1 Percent of Correct classification: PCC

We have measured the performance of our algorithm according to the PCC. A well known criteria which represents the percent of the correct classification. It is defined by:

$$PCC = \frac{\text{number of correct functional classified structures}}{\text{total number of classified structures}} \times 100 \tag{4.1}$$

For each classified instance, we compare its predicted class given by the k-nn classifier with its real functional class. Therefore, the total number of correctly classified instances is that of the instances whose class is correct

The more the PCC tends to 100%, the better our similarity measure based on the community concept is.

4.3.1.2 Validation criteria

We used our novel measure as a distance in the context of a k-nearest-neighbor classifier in a leave-one-out cross-validation setting.

In fact, in order to determine the class of a graph G_i we follow these following steps:

1. Compute the distance between G and all the other instances in the training set.
2. Retain the k nearest graphs that are closest to G.
3. The class of G is assigned on the basis of the majority classes relative to the k nearest neighbors.
4. Verify the correctness of the classification by checking whether the assigned class C_{pred} is the same as the actual class C_{real}.

These steps are repeated for all the data structures graphs. Then, the percentage of correct predictions is determined.

4.3.2 Experimentation on real datasets

4.3.2.1 Description of the datasets

To apply our proposed method to the comparison of protein binding sites structures, we considered a dataset of protein binding sites of two protein classes, ATP (Adenosine triphosphate) binding ligands and NADH (Nicotinamide adenine dinucleotide) binding ligands.

To obtain this dataset, we retrieved all ATP and NADH binding sites co-crystallized with each ligand using CavBase. Afterward, we reduced the collection to one cavity for each protein. Hence, we represented each one by a single binding pocket. This datatset is challenging since ATP is a substructure of NADH, it may possibly bind to the same ligands (see Appendix A for more information). It consists of 112 protein binding sites in the .rlbcoor format distributed in the following way (Table 4.1):

Table 4.1: Description of the database

Functional family	Number of instances
ATP	54
NADH	58

4.3.2.2 Experimental results

In the experimental phase, we have performed tests using the data set described above on an i5-3470 CPU (3.20 GHz) with 4 GB of RAM running Windows 8 Professional operating system. The experiments have been performed for three cases, according to the value of the clique, quasi-clique and the community similarity measures.

1. Clique similarity measure: In the first case, we test the efficiency of the standard clique detection to assess the similarity between a pair of graphs. We have implemented the standard clique detection using the Bron-Kerbosch algorithm (Bron & Kerbosch, 1973).

2. Quasi-clique similarity measure: In the second case, we tested the quasi-clique similarity measure using two methods, the one proposed in (Fober et al., 2013) which we call the merging-based method and the one proposed in (Boukhris et al., 2009) which we call the pruning-based method, according to $\gamma = 0.5$. We selected these two methods because they have been applied in the context of comparing protein binding site for protein function prediction. Indeed, testing them is substantial because the quasi-clique allows a degree of flexibility in the context of graph matching.

3. Community similarity measure: In the third case, we tested the efficiency of the community detection method in measuring the similarity between two graphs using the maximum densest community measure.

The parameters specification required (for the three cases) are the edges threshold (ϵ in Equation 2.4) for the product graph construction, which we set to 0.2 since it has proved to be well-suited (Fober et al., 2008), and the φ parameter for the similarity computation, which defines the type of comparison and interpolates between a measure of generalization and a measure of inclusion. φ is set to 0.5 (Equation 3.4) in order to balance between pure equivalence and pure inclusion.

Classification results

The results on the classification experiments of the four methods in terms of accuracy are summarized in Table 4.2.

Table 4.2: Accuracy of the four methods

Method	Accuracy		
k-nn	$k = 1$	$k = 3$	$k = 5$
Clique measure	49.10%	58.03%	58.03%
Quasi-clique measure[1]	69.64%	73.21%	71.42%
Quasi-clique measure[2]	71.42%	75.89%	76.78%
Community measure	76.78%	81.25%	79.46%

As it can be seen, the performance of our novel method outperformed the clique and quasi-clique detection approaches. It is able to detect similar graphs on the basis of the largest dense module that we have considered. Compared to the highest result of the

[1]Merging-based method
[2]Pruning-based method

clique-measure (58.03% for k=3 and k=5) and the quasi-clique measure (73.21% for k=3 for the first method[1] and 76.78% for k=5 for the second method[2]), the best performance of our maximum densest community measure (81.25% for $k = 3$) is higher as shown in Figure 4.1. These results confirm our conjecture that a relaxation of the *mcs* can increase the classification rate.

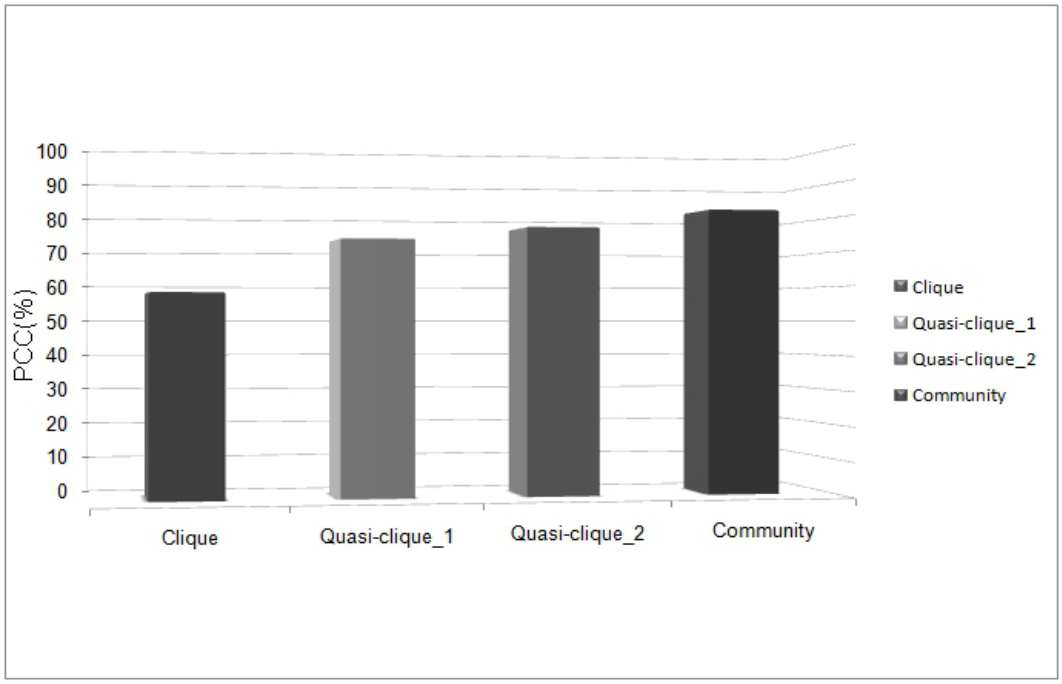

Figure 4.1: Accuracy of the four methods

Indeed, tolerance toward mismatches and minor differences that occur in graphs representing real world data are required especially in the case of biological data due to conformational flexibility and measurement errors that may occur frequently.

Besides, our method outperformed significantly these methods in terms of running time, it takes less than a minute to extract the maximum densest community between a pair of graphs, especially compared with the quasi-clique detection methods which are extremely time expensive and are ineffective on large graphs.

Although, our method is much better than the other methods in terms of time complexity. Indeed, the time complexity of the four methods are as follows (Table 4.3):

[1]Merging-based method
[2]Pruning-based method

Table 4.3: Time complexity of the four methods

Method	Time complexity
Maximum clique detection	$O(3^n)$
Maximum quasi-clique detection[1]	$O(2^n)$
Maximum quasi-clique detection[2]	$O(2^n)$
MDCC detection	$O(nlog(n))$

Where n is the number of nodes of the product graph.

As it can be seen, our method is much less costly in terms of time complexity compared to the other methods $O(nlog(n)) \ll O(2^n) < O(3^n)$.

4.4 Conclusion

In this chapter, we have presented the major parameters, programs and algorithms that we have implemented in order to test the efficiency and feasibility of our novel method. We also presented the experimental results obtained from simulation on real datasets of protein binding sites.

The experimental results show that our new approach gives better results than both the standard clique method and quasi-clique methods in terms of accuracy. These results confirm our conjecture, the community concept provides a more tolerant large common substructure that remains dense. Moreover, our method is really fast thanks to the community detection algorithm (Blondel et al., 2008) that we have applied unlike the other methods of cliques and quasi-clique which become ineffective on large graphs in terms of memory and time (Fober et al., 2013).

[1]Merging-based method
[2]Pruning-based method

General conclusion and future works

In this master thesis, we have provided a new method for assessing similarities between node-labeled and edge-weighted graphs representing protein binding sites based on the community concept. Our approach measures the structural correspondence between proteins independently of sequence and fold information in order to predict their functional class.

Since the protein structure is more closely to function than its sequence, we use graphs as models to represent protein binding sites structures. Indeed, graphs proved their effectiveness in modeling objects without information loss.

We predict the family class to which a protein belongs to using a graph matching technique that compares a pair of graphs and allows their structural analysis by measuring their similarity according to their common substructure. Our aim is to optimize this common substructure size while keeping it biologically and geometrically meaningful.

Existing methods assess similarities between graphs using the maximum clique extracted from the product graph. However, the clique concept is not appropriate in the context of comparing graphs of real world data because the latter is frequently affected by noise and distortions. Other methods use the γ-quasi-clique concept where the value of γ provides a relaxation to the clique notion by fixing a threshold to the degree of vertices. However, the problem of detecting the maximum γ-clique is an NP-complete problem since the downward closure property does not hold for quasi-cliques, which makes its use inefficient in the context of large graphs. Our idea, is to use another concept that does not require constraints on vertices connectedness named community which is a dense subgraph that can be considered as a generalization of the quasi-clique concept and that is well adapted for very large graphs.

In order to show the validity of our graph-based similarity measure, we have performed

experiments on real databases. We have evaluated it using the PCC. The experimental results showed that our approach is effective. The similarity between graphs based on maximum densest community concept outperforms both the standard maximum clique detection method and the maximum quasi-clique detection method. Besides, it is able to detect the maximum "almost" common subgraph in short time unlike the other methods which are time consuming. The major results will appear in (Mallek et al., 2014).

An interesting future work is to produce a graph that captures the common and frequent regions between a number of our maximum almost common subgraphs extracted from protein binding sites of the same functional family. In fact, this task can be carried out by choosing a graph and comparing it with the other graphs while keeping nodes and edges that are conserved in almost comparisons which is the concept of graph alignment methods. This graph can be used as a prototype for classification to better classify proteins. Besides, it would be interesting to apply our approach in other real world applications such as chemical compounds analysis. Finally, one should consider broader databases that include a larger number of instances.

Appendix

Appendix A

NADH-ATP dataset

A.1 Introduction

Molecular interactions of a protein with other molecules take place at the level of its binding sites. Therefore, analysis of protein binding pockets is very important to understand molecular interactions and predict the biochemical function a protein performs. Molecular recognition is established due the complementarity in shape and physicochemical properties between the binding site and the binding molecule.

From this point of view, approaches based on structure similarity using binding sites have been developed in the last years. Tools are provided to detect protein binding sites and to store them into data sources such as CavBase, a database for the fully automated extraction and storage of putative protein binding sites from proteins structure determined experimentally deposited in the PDB database (Berman et al., 2000). Since we are interested in the prediction of a protein function using its binding pockets we used a two-class dataset: Adenosine-triphosphate (ATP)- and nicotine amide dinucleotide (NADH)-binding proteins. Each of the classes share a prominent co-factor and will be detailed in this appendix.

A.2 Nicotinamide adenine dinucleotide -NADH

NADH is a coenzyme invovled in redox reactions, it is composed of nicotinamide adenine dinucleotide (NAD) and hydrogen (H) as shown in Figure A.1. The NADH molecule carries electrons causing interactions. Its role is very important since it helps enzyme-catalysed

reactions.

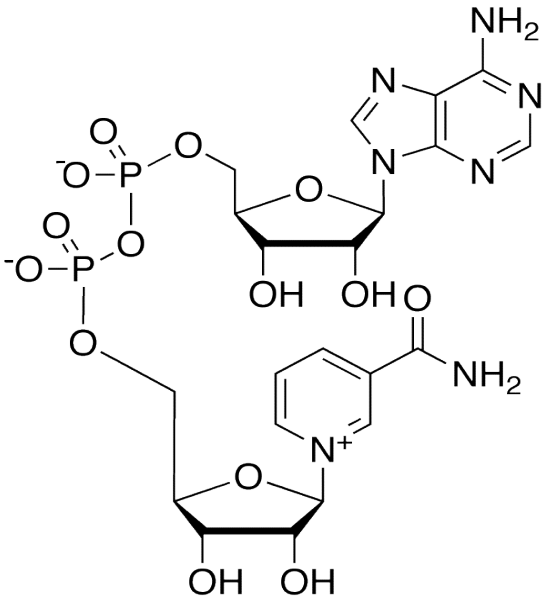

Figure A.1: Nicotinamide adenine dinucleotide -NADH

A.3 Adenosine triphosphate -ATP

Known as the "molecular unit of currency" of life. ATP is a coenzyme responsible for energy transfer. It is composed of adenine, ribose and three phosphate groups as shown in Figure A.2. The ATP molecule produces the majority of cellular energy ($\simeq 95\%$).

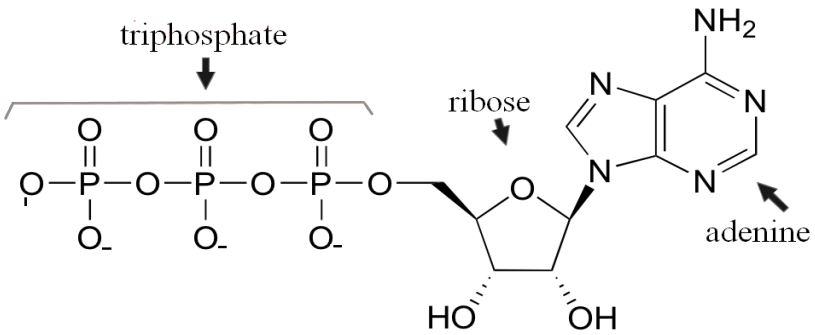

Figure A.2: Adenosine triphosphate- ATP

A.4 Conclusion

In this appendix, we have described some information about the ATP- and NADH-binding proteins. We have used this dataset in our experiments in order to test our method.

References

Abello, J., Resende, M., & Sudarsky, S. (2002). Massive quasi-clique detection. In *Proceedings of the 5th latin american symposium on theoretical informatics* (p. 598-612). Springer-Verlag.

Agarwal, S. K. (2008). *Bioinformatics*. S.B Nangia APH Publishing Corporation.

Altschul, S., Gish, F., Meyers, W. M. E., & Lipman, D. J. (1990). Basic local alignment search tool. *J Mol Biol*, *3*, 403-410.

Altschul, S., Madden, T. L., Schffer, A., Zhang, J., Miller, W., & Lipman, D. J. (1997). Psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, *17*, 3389-3402.

Alvareza, M., & Yanb, C. (2012). A new protein graph model for function prediction. *Computational Biology and Chemistry*, *37*, 6-10.

Andreeva, A., Howorth, D., Brenner, S. E., Hubbard, T. J. P., Chothia, C., & Murzin, A. G. (2004). Scop database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Research*, *32*, 226-229.

Balachander, K., & Jia, W. (2000, August). On network-aware clustering of web clients. *SIGCOMM Comput. Commun. Rev.*, *30*(4), 97-110.

Berman, H. M. (2013). *Pdb annual report 2013* (Tech. Rep.). Rutgers, The State University of New Jersey, University of California: Reseach collaboratory for structural bioinformatics.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., & Weissig. (2000). The protein data bank. *Nucleic Acids Res*, *28*(1), 235-242.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of community hierarchies in large networks. *CoRR*, *abs/0803.0476*.

Boukhris, I., Elouedi, Z., Fober, T., Mernberger, M., & Hüllermeier, E. (2009). Similarity analysis of protein binding sites: A generalization of the maximum common subgraph measure based on quasi-clique detection. In *Isda* (p. 1245-1250). IEEE Computer

Society.

Bron, C., & Kerbosch, J. (1973). Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, *16*(9), 575-577.

Brylinski, M., & Skolnick, J. (2008). A threading-based method (findsite) for lingand-binding site prediction and functional annotation. *Proc. Natl. Acat. Sci. USA*, *38*, 129-134.

Bunke, H. (2000). Recent developments in graph matching. In *Icpr* (p. 2117-2124).

Bunke, H., Jiang, X., & Kandel, A. (2000). On the minimum common supergraph of two graphs. *Computing*, *65*(1), 13-25.

Campbell, N. A., & Reece, J. B. (2008). *Biology.* Benjamin-Cummings Publishing Company.

Chakrabarti, S., Venkatramanan, K., & Sowdhamini, R. (2003). SMoS: a database of structural motifs of protein superfamilies. *Protein Engineering*, *16*(11), 791-793.

Charalampos, T., Francesco, B., Aristides, G., Francesco, G., & Maria, T. (2013). Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining* (p. 104-112). New York, NY, USA: ACM.

Chen, M., & Hofestadt, R. (2006). A new method to detect related function among proteins independent of sequence and fold homology. *Journal of Biomedical Informatics*, *39*, 147-159.

Chua, H. N., Sung, W.-K., & Wong, L. (2007). Using indirect protein interactions for the prediction of gene ontology functions. *BMC Bioinformatics*, *8*(S-4).

Daxin, J., & Jian, P. (2009). Mining frequent cross-graph quasi-cliques. *ACM Trans. Knowl. Discov. Data*, *2*(4), 16:1-16:42.

Ferre, F., Aussiello, G., Zanzoni, A., & Helmer-Citterich, M. (2004). Surface: a database of protein surface regions for functional annotation. *Nucleic Acids Research*, *32*, 989 - 1000.

Ferrer, M., Valvenya, E., & Serratosa, F. (2009). Median graph: A new exact algorithm using a distance based on the maximum common subgraph. *Pattern Recognition Letters*, *30*, 579-588.

Fober, T., Hüllermeier, E., & Mernberger, M. (2008). Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules. In A. Beyer & M. Schroeder (Eds.), *German conference on bioinformatics* (Vol. 136, p. 44-53). GI.

Fober, T., Klebe, G., & Hüllermeier, E. (2013). Local clique merging: An extension of the maximum common subgraph measure with applications in structural bioinformatics. In *Algorithms from and for nature and life* (p. 279-286). Springer.

Fober, T., Mernberger, M., Moritz, R., & Hullermeier, E. (2009). Graph-kernels for the comparative analysis of protein active sites. *GI*, *157*, 21-31.

Fornes, O., Garcia-Garcia, J., Bonet, J., & Oliva, B. (2014). Chapter four - on the use of

knowledge-based potentials for the evaluation of models of protein-protein, protein-dna, and protein-rna interactions. In R. Donev (Ed.), *Advances in protein chemistry and structural biology* (Vol. 94, p. 77-120). Academic Press.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*, 75-174.

Fuller, J. C., Burgoyne, N. J., & Jackson, R. M. (2009). Predicting druggable binding sites at the protein-protein interface. *Drug Discovery Today*, *14*(3-4), 155-161.

Gardiner, E. J., Artymiuk, P. J., & Willett, P. (1997). Clique-detection algorithms for matching three-dimensional molecular structures. *Journal of Molecular Graphics and Modelling*, *15*(4), 245-253.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of np-completeness.* W H Freeman and Co.

Gibrat, J., Madej, T., & Bryant, S. (1996). Surprising similarities in structure comparison. *Struct. Biol.*, *6*, 377-385.

Gursoy, A., Keskin, O., & Nussinov, R. (2008). Topological properties of protein interaction networks from a structural perspectives. *Biochemical Society Transactions*, *36*, 1398-1403.

Guruprasad, K., Prasad, M. S., & Kumar, G. R. (2000). Database of structural motifs in proteins. *Bioinformatics*, *16*(4), 372-375.

Håstad, J. (1997). Clique is hard to approximate within $n^{1\text{-epsilon}}$. *Electronic Colloquium on Computational Complexity (ECCC)*, *4*(38), 379-386.

Hendlich, M., Rippmann, F., & Barnickel, G. (1997). Ligsite: automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of Molecular Graphics and Modelling*, *15*(6), 359 - 363.

Holm, L., & Sander, C. (1995). Dali: a network tool for protein structure comparison. *Biochem Sci*, *11*, 478-80.

Horst, B., & Kim, S. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, *19*(3-4), 255-259.

Huang, B. (2009). Metapocket: A meta approach to improve protein ligand binding site prediction. *Integrative Biology*, *13*(4).

Jang, Y. S., Park, J. M., Choi, S., Choi, Y. J., Seung, D. Y., Hee, J. C., & Lee, S. Y. (2012). Engineering of microorganisms for the production of biofuels and perspectives based on systems metabolic engineering approaches. *Biotechnology Advances*, *30*(5), 989 - 1000. doi: http://dx.doi.org/10.1016/j.biotechadv.2011.08.015

Jones, D., & Hadley, C. (1999). A systematic comparison of protein structure classifications:scop, cath and fssp. *Structure*, *7*, 21-35.

Kahraman, A., Morris, R. J., Laskowski, R. A., & Thornton, J. M. (2007). Shape variation in protein binding pockets and their ligands. *Journal of Molecular Biology*, *368*(1), 283-301.

Keedwell, E., & Narayanan, A. (2005). *Intelligent bioinformatics : The application of artificial intelligence techniques to bioinformatics problems.* John Wiley and Sons.

Kinnings, S., Liu, N., Buchmeier, N., Tonge, P., Xie, L., & Bourne, P. (2009). Drug discovery using chemical systems biology: Repositioning the safe medicine comtan to treat multi-drug and extensively drug resistant tuberculosis. *PLoS Computational Biology*, *5*(7), e1000423+.

Klebe, G., Hulemeirer, E., Weskamp, N., & Khun, D. (2007). Functional classification of protein kinase binding sites using cavbase. *ChemMedChem*, *2*, 1432-1447.

Li, X. L., Tan, S. H., Foo, C. S., & Ng, S. K. (2005). Interaction graph mining for protein complexes using local clique merging. *Xiao Li Li*, *16*(26), 260-269.

Liu, G., & Limsoon, W. (2008). Effective Pruning Techniques for Mining Quasi-Cliques. In *Proceedings of the european conference on machine learning and knowledge discovery in databases - part II* (p. 33-49). Springer-Verlag.

Lopez-Presa, J. L., & Anta, A. F. (2011). Fast isomorphism testing of graphs with regularly-connected components. *CoRR*, *1106.4489*.

Maiolica, A., Junger, M. A., Ezkurdia, I., & Aebersold, R. (2012). Targeted proteome investigation via selected reaction monitoring mass spectrometry. *Journal of Proteomics*, *75*(12), 3495-3513.

Mallek, S., Boukhris, I., & Elouedi, Z. (2014). Predicting proteins functional family: A graph-based similarity derived from community detection. In *Ieee international conference on intelligent systems (is'14).* Springer.

McKay, B. D. (1981). Practical graph isomorphism. *Congressus Numerantium*, *30*, 45-87.

McKay, B. D., & Piperno, A. (2014). Practical graph isomorphism II. *Journal of Symbolic Computation*, *60*(0), 94-112.

Newman, M. E. J., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review*, *69*(026113).

Nisius, B., Sha, F., & Gohlke, H. (2012). Structure-based computational analysis of protein binding sites for function and druggability prediction. *Journal of Biotechnology*, *159*(3), 123-134.

Niu, D.-K., & Jiang, L. (2013). Can ENCODE tell us how much junk DNA we carry in our genome? *Biochemical and Biophysical Research Communications*, *430*(4), 1340-1343.

Orengo, C. A., Bray, J. E., Bychan, D., Harrison, W. A., Lee, D., Pearl, F. M., & Silliote, G. (2002). The cath protein family database: a resource for structural and functional annotation of genomes. proteomics. *Proteomics*, *2*, 11-21.

Pattillo, J., Veremyev, A., Butenko, S., & Boginski, V. (2013). On the maximum quasi-clique problem. *Discrete Applied Mathematics*, *161*(1-2), 244-257.

Pearson, W. R., & Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proc Natl Acad Sci U.S.A*, *85*(8), 2444-2448.

Pérez, A. P., Dominguez-Sal, D., & Larriba-Pey, J.-L. (2014). High quality, scalable and parallel community detection for large real graphs. In *Proceedings of the 23rd international conference on world wide web* (p. 225-236). International World Wide Web Conferences Steering Committee.

PÃ©rot, S., Sperandio, O., Miteva, M. A., Camproux, A.-C., & Villoutreix, B. O. (2010). Druggable pockets and binding site centric chemical space: a paradigm shift in drug discovery. *Drug Discovery Today*, *15*(15-16), 656-667.

Raymond, J. W., Gardiner, E. J., & Willett, P. (2002). Heuristics for similarity searching of chemical graphs using a maximum common edge subgraph algorithm. *Journal of Chemical Information and Computer Sciences*, *42*(2), 305-316.

Raymond, J. W., & Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, *16*, 521-533.

Rost, B. (1999). Twilight zone of protein sequence alignments. *Protein Eng*, *12*, 85-94.

Rost, B., Nair, R., Wrzeszczynski, K. O., & Ofran, Y. (2003). Automatic prediction of protein function. *Cell Mol Life Sc*, *60*, 2637-2650.

Schmitt, S., Kuhn, D., & Klebe, G. (2002). A new method to detect related function among proteins independent of sequence and fold homology. *J. Mol. Biol.*, *323*, 387-406.

Schwikowski, B., Uetz, P., & Fields, S. (2000). A network of protein-protein interactions in yeast. *Nat Biotechnol*, *12*, 1257-61.

Scott, L. P., Barioni, M., & Stelle, D. (2011). Using data mining to identify structural rules in proteins. *Biophys*, *218*, 1997-2004.

SeeKiong, N. (2004). *Practical-bioinformatician*. Institute for Infocomm Research: Serigraph.

Skolnick, J., Fetrow, S., & Kolinski, A. (2000). Structural genomics and its importance for gene function analysis. *Nat. Biotechnol*, *18*, 283-287.

Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *J. ACM*, *23*(1), 31-42.

Vazquez, A., Flammini, A., Maritan, A., & Vespignani, A. (2003). Global protein function prediction from protein-protein interaction networks. *Nat Biotechnology*, *21*, 697-700.

Weskamp, N., Hullermeier, E., Kuhn, D., & Klebe, G. (2007, April). Multiple graph alignment for the structural analysis of protein active sites. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, *4*(2), 310–320.

Weskamp, N., Kuhn, D., Hullermeier, E., & Klebe, G. (2004). Efficient similarity search in protein structure databases by k-clique hashing. *Bioinformatics*, *20*(10), 1522-1526.

Wittig, U., Kania, R., Bittkowski, M., Wetsch, E., Shi, L., & Jong, L. (2014). Data extraction for the reaction kinetics database sabio-rk. *Perspectives in Science*.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, *18*(1), 183-190.

Yingjie, L., Seungyeul, Y., & Roberto, S. (2012). Sitecomp: a server for ligand binding site analysis in protein structures. *Bioinformatics*, *28*(8), 1172-1173.

Yuan, T. Y., Jeffrey, C. Z., & Hsiung, L. W. (2010). fpop: footprinting functional pockets of proteins by comparative spatial patterns. *Nucleic Acids Research*, *38*(Database-Issue), 288-295.

Zhang, S. Y., & Liu, S. L. (2013). Bioinformatics. In M. Stanley & H. Kelly (Eds.), *Brenner's encyclopedia of genetics (second edition)* (Second Edition ed., p. 338-340). San Diego: Academic Press.

Abstract

In this master thesis, we propose a method for predicting the functional family of proteins, independently of sequence or fold homologies, using similarities of their binding pockets. We use graphs as models to depict the geometry and physicochemical composition of these binding pockets without loss of information. Graph matching techniques are appropriate to compare two graphs and compute their similarities according to the size of their largest common substructure (the maximum common subgraph (*mcs*)). The latter is an overly stringent measure which is sensitive toward small deviations and errors. Besides, its mining is difficult especially on large graphs. In order to overcome these issues, we propose a relaxation of the *mcs* measure based on so-called communities. A community is used as an "almost common" subgraph with high concentrations of edges. With our approach, we increase tolerance towards noise and structural variation and we detect the largest "almost" common substructure with low computation time. The proposed measure is validated by an experimental study in order to check the impact of our new measure on the similarity assessment and functional classification of proteins.

Keywords: Graph-based similarity, Community, Protein binding pockets classification, Maximum almost common subgraph, Functional classification.

Résumé

Dans ce mémoire de mastère, nous proposons une méthode pour la prédiction de la famille fonctionnelle des protéines, indépendamment des homologies de séquence ou de repliement, en comparant les similarités de leurs poches de liaison. Nous utilisons les graphes comme modèles afin de représenter la géométrie et la composition physicochimique de ces poches de liaison sans perte d'informations. Les techniques d'appariement qui ustilisent la taille de la plus grande sous-structure commune (le maximum sous-graphe commun (*mcs*)) sont appropriées pour effectuer la comparaison et calculer les similarités entre les graphes. Cependant, le *mcs* est difficile à détecter sur des graphes de grandes tailles. De plus, c'est une mesure très rigoureuse qui est sensible aux petits écarts et erreurs. Afin de résoudre ces problèmes, nous proposons une relaxation basée sur les communautés. Une communauté est utilisée comme un sous-graphe "presque" commun avec des concentrations élevées d'arÃªtes. Avec notre approche, nous augmentons la tolérance au bruit et les variations structurelles et nous réduisons le temps de calcul. La mesure proposée est validée par une étude expérimentale afin de vérifier son impact sur le calcul de similarités et la classification fonctionnelle des protéines.

Mots clés: Similarité basée sur les graphes, Communauté, Classification des poches de liaison des protéines, Sous-graphe maximum commun, Classification fonctionnelle.