2013

FAC-PIN: An efficient and fast agglomerative clustering algorithm for protein interaction networks to predict protein complexes and functional modules

Mohammad Shamsur Rahman
University of Windsor

FAC-PIN: AN EFFICIENT AND FAST AGGLOMERATIVE CLUSTERING ALGORITHM FOR PROTEIN INTERACTION NETWORKS TO PREDICT PROTEIN COMPLEXES AND FUNCTIONAL MODULES.

by

MOHAMMAD SHAMSUR RAHMAN

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2013

FAC-PIN: AN EFFICIENT AND FAST AGGLOMERATIVE CLUSTERING ALGORITHM FOR PROTEIN INTERACTION NETWORKS TO PREDICT PROTEIN COMPLEXES AND FUNCTIONAL MODULES.

by

MOHAMMAD SHAMSUR RAHMAN

APPROVED BY:

Dr. Abdulkadir Hussein, External Reader
Department of Mathematics and Statistics

Dr. Luis Rueda, Internal Reader
School of Computer Science

Dr. Alioune Ngom, Advisor
School of Computer Science

Dr. Christie Ezeife, Chair of Defense
School of Computer Science

May, 02, 2013

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Proteins are known to interact with each other to perform specific living organism functions by forming functional modules or protein complexes. Many community detection methods have been devised for the discovery of functional modules or protein complexes in protein interaction networks. One common problem in current agglomerative community detection approaches is that vertices with just one neighbor are often classified as separated clusters, which does not make sense for module or complex identification. In this thesis, we propose a new agglomerative algorithm, FAC-PIN, based on a local premetric of relative vertex-to-vertex clustering value. Our proposed FAC-PIN method is applied to PINs from different species for validating functional modules and protein complexes generated from FAC-PIN with experimentally verified functional modules and complexes respectively. The preliminary computational results show that FAC-PIN can discover functional modules and protein complexes from PINs more accurately. As well as we have also compared the computational times for different species with HC-PIN and CNM algorithms. Our algorithm outperforms two algorithms. Our FAC-PIN algorithm is faster and accurate algorithm which is the current state-of-the-art agglomerative approach to complex prediction and functional module identification.

Dedication

I would like to dedicate this thesis to my creator and Everlasting true Almighty Allah. My parents and wife always want to see me reaching the highest levels of success. I am very happy today that I am able to make them proud. Also thanks to my parent and wife for all their prayers. It is because of their good wishes, I am able to do this research work.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to Dr. Alioune Ngom, my supervisor, for his steady encouragement, patient guidance and enlightening discussions throughout my graduate studies. Without his help, the work presented here would have not been possible. Dr. Alioune Ngom will always remain like a father figure to me. I also wish to express my appreciation to Dr. Christie Ezeife, Dr. Luis Rueda, School of Computer Science and Dr. Abdulkadir Hussein, Department of Mathematics and Statistics for being in the committee and spending their valuable time.

Contents

Author's Declaration of Originality **iii**

Abstract **iv**

Dedication **v**

Acknowledgements **vi**

List of Figures **x**

List of Tables **xii**

1 Introduction **1**

1.1 Background 1

1.1.1 Protein 1

1.1.2 Protein complex and Functional Modules 4

1.1.3 Protein Interaction Networks 5

1.1.4 Community 7

1.1.5 Community Detection Algorithm 10

1.2 Objectives 12

1.3 Thesis Organization 13

2 Relative Works 15

2.1 Density based methods 15

2.1.1 Spirin et al 2003 16

2.1.2 Li et al. 2006 16

2.1.3 Altaf et al. 2006 17

2.1.4 Pei et al. 2007 18

2.1.5 Summary 19

2.2 Graph partitioning approaches 20

2.2.1 Dongen 2000 20

2.2.2 King et al. 2004 20

2.2.3 Graph Entropy Algorithm 21

2.2.4 Summary 22

2.3 Hierarchical based methods 22

2.3.1 Ravasz et al. 2002 23

2.3.2 Girvan et al. 2002 23

2.3.3 Newman 2003 24

2.3.4 Radicchi et al 2004 25

2.3.5 Clauset et al 2004 26

2.3.6 Luo et al 2007 27

2.3.7 Li et al. 2008 27

2.3.8 Wang et al 2011 28

2.3.9 Summary 29

3 Fast Agglomerative Clustering Algorithms 31

3.1 Relative Vertex-to-Vertex Clustering Value 31

3.2 The FAC-PIN Algorithm 35

3.2.1 The algorithm 35

3.2.2 Computational Complexity 35

4 Computation experiments, results and discussions 39

4.1 Datasets 40

4.2 Testing by Scoring Functions 41

4.3 Protein Complex Discovery 47

4.4 Functional Module Identification 55

4.5 Conclusion 59

5 Conclusion and future work 63

Bibliography 66

Vita Auctoris 71

List of Figures

1.1 Levels of protein structures [28] 2

1.2 α helix and β sheet of secondary structure of proteins [3] 3

1.3 Protein complexes of Baker's yeast [15] 5

1.4 Hyperclique pattern of functional modules in a protein complex [32] 6

1.5 Protein Interaction Network of Baker's yeast [14] 6

1.6 Community structure of a graph *G* 8

1.7 Overlapping and non-overlapping communities of a graph *G* [23] 9

1.8 Scale Free Network *G* [9] 11

4.1 Modularity comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog 43

4.2 Modularity comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish 43

4.3 -(w-log-v) comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog 44

4.4 -(w- log -v) comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish 44
4.5 Log-log plot of execution time comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog 45
4.6 Log-log plot of execution time comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish 46
4.7 Protein Interaction Network of Baker's yeast (Y_{4K}) 48
4.8 Clustered Network of Baker's yeast (Y_{4K}) using FAC-PIN algorithm 49
4.9 Clustered Network of Baker's yeast (Y_{4K}) using CNM algorithm 50
4.10 Clustered Network of Baker's yeast (Y_{4K}) using HC-PIN algorithm 51

List of Tables

2.1 Summary of previous algorithms based on density based method 19

2.2 Summary of previous algorithms based on graph partitioning method 22

2.3 Summary of previous algorithms based on hierarchical method 29

4.1 Dataset used in experiments 40

4.2 *Q* results of *FAC-PIN*, *CNM* and *HC-PIN* for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog. 42

4.3 *Q* results of *FAC-PIN*, *CNM* and *HC-PIN* for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish. 42

4.4 $-(w\text{-}\log\text{-})v$ results of *FAC-PIN*, *CNM* and *HC-PIN* for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog. 42

4.5 $-(w\text{-}\log\text{-}v)$ results of *FAC-PIN*, *CNM* and *HC-PIN* for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish. 42

4.6 Time comparison results of *FAC-PIN*, *CNM* and *HC-PIN* for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog (in seconds). 47

4.7 Time comparison results of *FAC-PIN*, *CNM* and *HC-PIN* for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish (in seconds). 47

4.8 Comparison of the *Specificity*, *Sensitivity* and *F-score FAC-PIN*, *CNM* and *HC-PIN* for *Baker's yeast* 54

4.9 Comparison of the *Specificity*, *Sensitivity* and *F-score FAC-PIN*, *CNM* and *HC-PIN* for *Human* 54

4.10 Comparison of the *Specificity*, *Sensitivity* and *F-score FAC-PIN*, *CNM* and *HC-PIN* for *Mouse* 54

4.11 Comparison of the *Specificity*, *Sensitivity* and *F-score FAC-PIN*, *CNM* and *HC-PIN* for *Rat* 55

4.12 Functional enrichment of the identified Modules which comprises of three or more proteins of *Baker's yeast* 58

4.13 Performance comparison of clustering algorithms for the identification of large-sized ($size \geq 20$) modules of *Baker's yeast* 58

4.14 Performance comparison of clustering algorithms for the identification of small-sized ($size \leq 6$) modules of *Baker's yeast* 59

4.15 Functional enrichment of the identified Modules which comprises of three or more proteins of *Rat* 60

4.16 Performance comparison of clustering algorithms for the identification of large-sized ($size \geq 20$) modules of *Rat* 60

4.17 Performance comparison of clustering algorithms for the identification of small-sized ($size \leq 6$) modules of *Rat* 60

Chapter 1

Introduction

In the current Chapter, we discuss necessary background and the objective of the thesis. In Section 1.1, we describe the important terminologies which help to understand the objectives of the thesis. In Section 1.2, we have described the objectives of the thesis in detail. Finally in Section 1.3, we have shown the organization of the thesis.

1.1 Background

1.1.1 Protein

Proteins are large biological molecules consisting of multiple chains of amino acids [1] [28]. Proteins perform a vast array of functions within living organisms, including catalyzing metabolic reactions, replicating DNA, responding to stimuli, and transporting molecules from one location to another. Proteins were first described by the Dutch chemist Gerardus Johannes Mulder and named by the Swedish chemist Jöns Jacob Berzelius in 1838.

[1]Amino acids are biologically important organic compounds made from amine (-NH2) and carboxylic acid (-COOH) functional groups, along with a side-chain specific to each amino acid

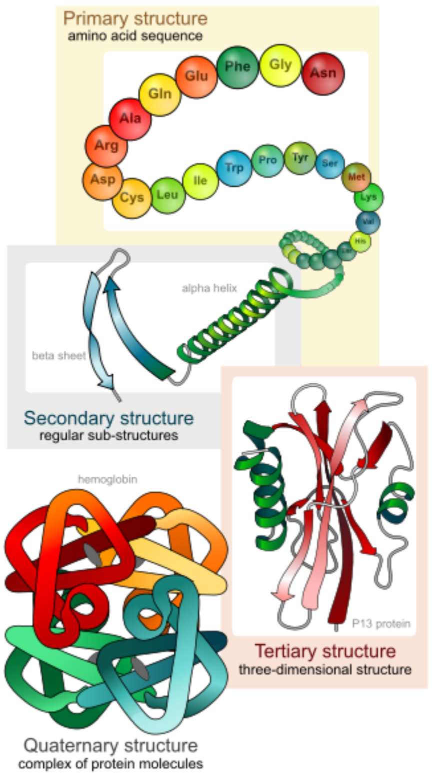

Figure 1.1: Levels of protein structures [28]

A protein having multiple chains of amino acids has four levels of structure [28]: The four levels of the protein structure are shown in Figure 1.1. ***Primary structure*** of a protein is the linear sequence of its amino acid structural units and partly comprises its overall biomolecular structure [28]. The primary structure is held together by covalent or peptide bonds, which are made during the process of protein biosynthesis[2]. The two ends of the polypeptide chain are referred to as the carboxyl terminus (C-terminus) and the amino terminus (N-terminus) based on the nature of the free group on each extremity. Counting of residues[3] always starts at the N-terminal end (-NH2 group) and ends at the C-terminal

[2]Biosynthesis is an enzyme-catalyzed process in cells of living organisms by which substrates are converted to more complex products [4]

[3]residue refers to a specific amino acid within the polymeric chain of a protein or nucleic acid

end (-COOH group). The sequence of a protein is unique to that protein, and defines the structure and function of the protein. The sequence of a protein can be determined by methods such as Edman degradation[4] or Tandem mass spectrometry[5] [28]. Often however, it is read directly from the sequence of the gene using the genetic code. There are more than thousands types of proteins in our body which are composed of different arrangements of 20 types of amino acid residues.

Secondary structure refers to highly regular local sub-structures. The secondary structure consists of two major strcutures: the alpha helix (are often the basis of fibrous polymers) and the beta strand or beta sheets (often has twists that increase the strength and rigidity of the structure), were suggested in 1951 by Linus Pauling and co-workers [24]. These secondary structures are defined by patterns of hydrogen bonds between the main-chain peptide groups. Both the alpha helix and the beta-sheet represent a way of saturating all the hydrogen bond donors and acceptors in the peptide backbone. α helix and β sheet of secondary structure of proteins are shown in Figure 1.2

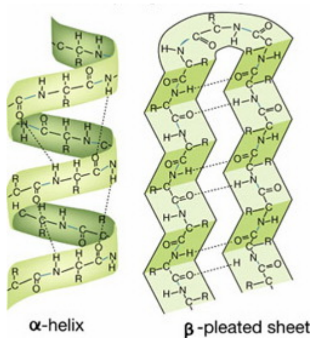

Figure 1.2: α helix and β sheet of secondary structure of proteins [3]

Tertiary structure of a protein is when the molecule is further folded and held in a particular complex shape forming precise and compact structure, unique to that protein.

[4]Edman degradation, developed by Pehr Edman, is a method of sequencing amino acids in a protein
[5]Tandem mass spectrometry refers to the application of mass spectrometry to the study of proteins

The shape is maintained permanently by the intra- molecular bonds:

Hydrogen bond of one hydrogen atom shared by two other atoms

Van der Waals force is the weak force that incurs when two or more atoms are very close

Disulphide bond is a strong covalent bond formed between two adjacent cysteine amino acids. The bond stabilizes the tertiary shape of a protein

Ionic bond is the electrostatic interaction between oppositely charged ions

Quaternary structure formed by several protein molecules (polypeptide chains), usually called protein subunits in this context, which function as a single protein complex. The quaternary structure is stabilized by the same non-covalent interactions and disulphide bonds as the tertiary structure. Quaternary structure of protein arise when a number of tertiary polypeptides joined together forming a complex or time to time modules, biologically active molecule.

1.1.2 Protein complex and Functional Modules

Protein complex are groups of proteins that interact with each other at the same time and place, forming a single multimolecular machine [29]. These are the form of quaternary structure of proteins. Identified protein complexes include several large transcription factor complexes, the anaphase-promoting complex, RNA splicing and polyadenylation machinery, protein export and transport complexes etc. Protein complexes of Baker's yeast is shown in Figure 1.3.

Functional modules are consisted of proteins that participate in a common elementary biological process while binding each other at a different time and place (different conditions or phases of the cell cycle, in different cellular compartments etc.) [29]. Example of

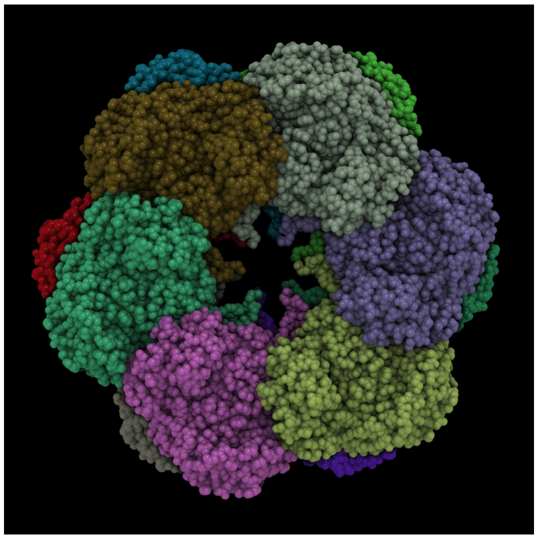

Figure 1.3: Protein complexes of Baker's yeast [15]

identified functional modules including the CDK/ cyclin module responsible for cell-cycle progression, the yeast pheromone response pathway, MAP signaling cascades etc. A 3D structural view of hyperclique pattern of functional modules within a protein complex is shown in Figure 1.4. It is very important to remember, functional modules contain multiple protein complexes [5, 10]. On the other hand, protein complexes carry out a specific task, but functional modules carry out a set of tasks which are carried out by individual protein complexes [10].

1.1.3 Protein Interaction Networks

Network representation of proteins and their interactions are known as ***P**rotein **I**nteraction **N**etwork* [29]. In short it is called *PIN*. In PINs, proteins are represented as nodes or vertices and interactions are as edges. Maximum PINs are undirected networks with edge weight or not [13, 29]. In Figure 1.5, an unweighted PIN of baker's yeast is shown.

Girvan and Newman [12] and Fortunato [9] discuss about the five properties of protein

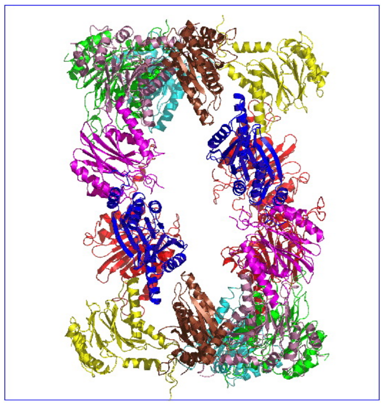

Figure 1.4: Hyperclique pattern of functional modules in a protein complex [32]

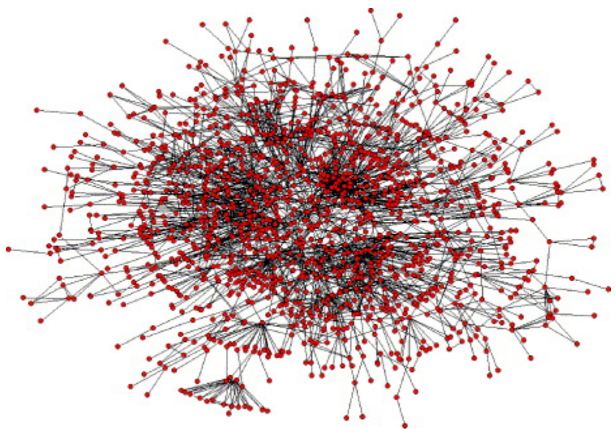

Figure 1.5: Protein Interaction Network of Baker's yeast [14]

interaction networks in their papers:

Small world effect which is the name given to the finding that the average distance between vertices in a network is small.

Power law degree distribution is a distribution where the number of the vertices with low degree is higher than the number of vertices with high degree.

Network transitivity is a property that two vertices that are both neighbor of same third vertex have a heightened probability of also being neighbor of one another.

Community structure is a property where intrales or both.-connectivity of a subset of vertices of graph G is higher than inter-connectivity between others. It is briefly discussed in Subsection 1.1.4.

Preferential attachment is a property where a new node u is likely to *attach* to a high-degree node v than to a low degree node.

In PINs, all protein complexes and functional modules are strong subgraphs[6] [29]. To identify the protein complexes or functional modules from PINs means strong subgraphs, authors of the algorithms were used any of five properties. Third and fourth properties are commonly used to discover protein complexes or functional modules. But unfortunately, fifth property have not still used by any authors which helps to identify the more significant strong subgraph having biological significance.

1.1.4 Community

A community is defined as a subgraph (a subset of vertices of graph G) within the graph G such that connections inside the subgraph are denser than connections with the rest of the network [26]. Luo et al [21] gave the more formal definition of community. Their definition is as followed-

Definition 1.1.1. Community U is a subgraph of a graph G in which in-degree of U is higher than out-degree and the ratio of in-degree and out-degree of U should be higher than 1.

In-degree of a community U is the number of edges connected between the vertices of community U and out-degree of a community U is the number of edges between other

[6]A subgraph has high concentration of edges

communities and U. From the formal definition of community, two properties of the community are revealed-

Homogeneity: Vertices of a community are highly similar or compact to each other.

Separability: Vertices of different communities have lower similarity or compactness.

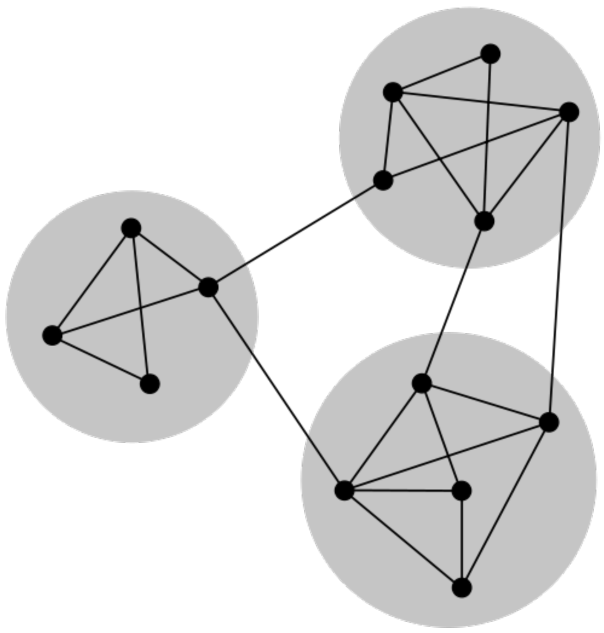
Figure 1.6: Community structure of a graph G

On the other hand, inhomogeneity or separability property suggests that the network has certain natural divisions within it. The communities are often defined in terms of the partition of the set of vertices, that is each node is put into either only one community just as in the Figure 1.6 or into multiple communities. Depends on the distribution of the nodes among the communities, community can be classified into two groups-

Overlapping communities share one or more common nodes among them. In Figure 1.7, yellow, green and purple colored communities are sharing red colored vertices. These communities are the examples of overlapping communities

Non-overlapping communities do not share any node between them. In Figure 1.7, blue and purple; blue and yellow colored communities do not share a single vertex between. So, these communities are the example of non-overlapping communities.

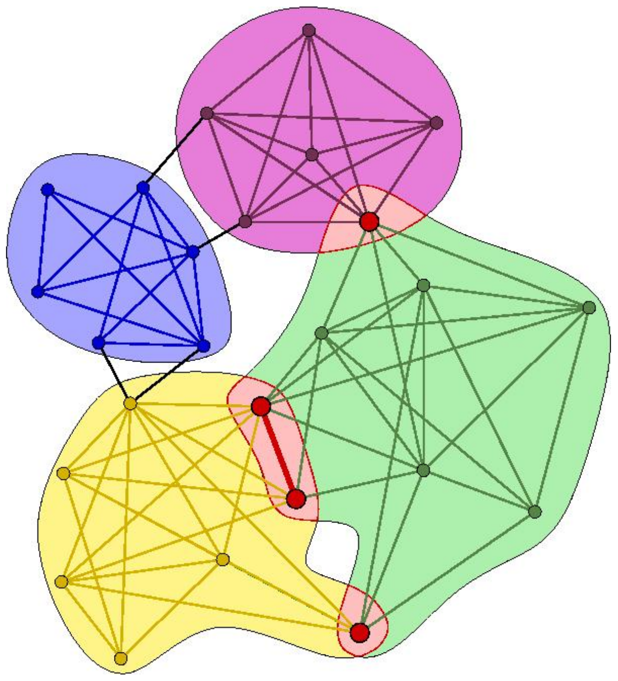

Figure 1.7: Overlapping and non-overlapping communities of a graph G [23]

In the Figure 1.7, blue and green colored communities are not connected by any edge. These communities are known as *disjoint* communities.

Moreover, Radicchi et al. [26] also classified the communities into two groups according their connectivities:

Strong community is a community U in which *in-degree*[7] of all vertices are higher than *out-degree*[8].

[7]The number of connected edges between internal vertices of community U.

[8]The number of connected edges between external vertices to community U.

Weak community is a community U in which in-degree of some vertices are higher than out-degree.

In PINs, protein complexes and functional modules are formed by interacting proteins. PINs organize into densely linked complexes where interactions appear with high concentration among the proteins of the complex [33]. It indicates the protein complexes or functional modules are the communities in PINs in respect to the network and community definition. Generally, in PINs, the number of interactions are very large than the number of proteins, like Figure 1.5. It is not easy and simple to identify the protein complexes or functional modules. Some computational methods are required for detecting protein complexes or functional modules from PINs. Community detection algorithms are very common to identify the complexes or modules from PINs.

1.1.5 Community Detection Algorithm

Community detection in PINs is a computationally hard task. Conventional clustering algorithms are not well suited for this task [25, 34]. Efficient, accurate, robust, and scalable methods are therefore required for mining large PINs. There are three approaches of community detection methods according to their working principles [9]:

Density based technique finds the subgraphs in the network whose density is higher [9]. But this method cannot find the communities or clusters efficiently for scale free networks[9], see Figure 1.8). Moreover all PINs are scale free networks. For this reason, density based algorithms are not used in clustering of PINs [9].

Graph partition techniques find the bridge edges which connect the communities. By

[9]A scale-free network is a network whose degree distribution follows a power law, at least asymptotically.

removing bridge edges, these algorithms discover the communities [8, 16]. These algorithms are very efficient, but suffered by execution time.

Hierarchical method finds the communities by calculating similarity or compactness between the nodes [9]. But this method cannot classify the vertices of degree one in same community with their neighbors which does not make sense biologically [9]. Time complexity is another problem of this method.

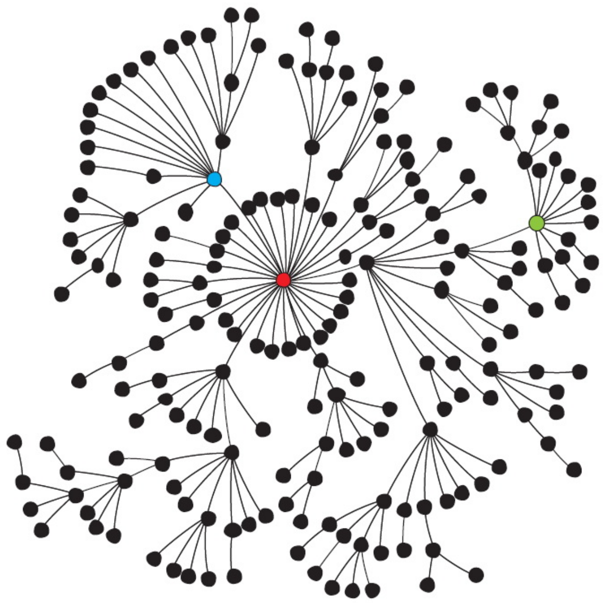
Figure 1.8: Scale Free Network *G* [9]

In this thesis, we put our emphasis on the problems of hierarchical method. We have designed a new algorithm which is known as *FAC-PIN* algorithm to solve the problems of hierarchical method.

1.2 Objectives

The objectives of the thesis are as follows:

To design a prematric to solve the problem of classifying vertices with one neighbor: In any hierarchical method, a metric or measure is used to cluster any PIN. But all proposed metric cannot solve the problem of clustering the vertices having degree one. In this thesis, we have proposed a new pre-metric[10] - *Relative vertex-to-vertex clustering value* which solves the problem of clustering vertices of degree one.

To design a hierarchical algorithm to improve the clustering processes for PINs: No hierarchical method can solve the problem of classifying the vertices of degree one. In this thesis, we have proposed a new agglomerative approach of hierarchical method to solve the problem of classifying nodes containing one neighbor by using *Relative vertex-to-vertex clustering value*. As well as our proposed algorithm has produced/ discovered more dense subgraphs in PINs than previous hierarchical algorithms.

To design a faster method for hierarchical approach: In 2011, Wang et al. [30] proposed a faster agglomerative hierarchical method for clustering PINs. The worst case time complexity of their algorithm is $O(\bar{d}^2 m)$ where m is the number of interaction and $\bar{d}$ is the average degree of any network G. It is the fastest algorithm so far published. On the other hand, we have proposed an agglomerative algorithm which is known as *FAC-PIN* algorithm. The worst case time complexity of FAC-PIN algorithm is $O(\bar{d}^2 n)$. In any protein interaction network, the number of proteins n is smaller than the number of interactions m.

[10]A metric satisfies axiom of positivity and axiom of positive definiteness

1.3 Thesis Organization

We organize our thesis into another four chapters. In Chapter 2, we discusse the previous works related to the clustering of PINs. We describe our proposed pre-metric *Relative vertex-to-vertex clustering values* and new agglomerative algorithm: *FAC-PIN* in the Chapter 3. After designing the algorithm, we carry out computation experiments on several PINs. We discuss the computation experiments and results in the Chapter 4. Finally in Chapter 5, we conclude our thesis with the discussion of FAC-PIN algorithms and its future works.

Chapter 2

Relative Works

In this Chapter, we discuss only the community detection algorithms which are directly involved in Protein interaction networks. All algorithms are designed on the definitions of the community structures. For PINs, community detection algorithms are classified into three groups according to their working principles: *Density* based methods, *Graph partitioning* based approaches and *Hierarchical* methods. In Section 2.1, we discuss the algorithms which are designed on the principle of the density of the subgraphs. We discuss the graph partitioning algorithms in Section 2.2 and hierarchical methods in Section 2.3.

2.1 Density based methods

All density based methods find the dense subgraph by several density measures or metrics (density functions, edge clustering coefficient, clustering property, network affinity, random walk etc.) The authors of the algorithms of this method introduced different density measure techniques to find the dense subgraphs in PIN. In the current Section, we discuss the density based algorithms which are only involved in protein interaction network clustering.

2.1.1 Spirin et al 2003

First Spirin et al. [29] designed an algorithm for finding protein complexes and functional modules from PINs based on density. In their method, at first it finds all cliques from a PIN. After finding all possible cliques[1], it uses the concept of Markov Cluatering algorithm (MCL which is discussed in Subsection 2.2.1) and Super Paramagnetic Clustering (SPC) for predicting dense subgraphs. In this method, MCL is used for identifying highly dense subgraph and SPC for predicting clusters that have very few connections to the rest of the network. The time complexity of the algorithm is $O(n^2k^2)$, where n and k are the number of vertices and the maximum size of cliques in the graph. It can find clusters of PIN efficiently by taking more time.

2.1.2 Li et al. 2006

Li et al [20] proposed a new method based on *Local clique merging* process in 2006. At first they have identified the local cliques in a PIN by using the *density of a subgraph.* They designed the equation of density of a subgraph based on *clustering coefficient*

$$cc(\acute{G}) = \frac{2\times \mid \acute{E} \mid}{\mid \acute{V} \mid \times \mid \acute{V} - 1 \mid} \tag{2.1}$$

where $\acute{G}$ is a subgraph of a graph G, $\acute{V}$ is subset of vertex set V and $\acute{E}$ is the subset of edge set E. After finding all cliques, their algorithm has merged the cliques to forms bigger dense subgraph by using *Neighborhood affinity* and a threshold ω. The neighborhood

[1]Clique in an undirected graph G = (V, E) is a subset of the vertex set $C \subseteq V$, such that for every two vertices in C, there exists an edge connecting the two

affinity follows following equation-

$$NA(A,B) = \frac{\mid A \cup B \mid^2}{\mid A \mid \times \mid B \mid} \tag{2.2}$$

Their algorithm is known as *Local Clique Merging Algorithm* or *LCMA*. The worst case time complexity of the algorithm is $O(lk^2v)$, where l, k and v are the number of iterations, the maximum size of local cliques and the average number of proteins in the local clique. LCMA can detect any size of complex except smaller size. But it suffers a common problem, that is- it classify the vertices of degree one are in different clusters from their neighbors.

2.1.3 Altaf et al. 2006

Altaf et al [1] has designed another density based clustering approach which solve one shortcoming (separating smaller cluster from larger one) of Li et al [20]. To do this, they have introduced new definition of density which is as follows-

$$d_k = \frac{\mid \acute{E} \mid}{\mid \acute{E} \mid_{max}} \tag{2.3}$$

Where $\mid \acute{E} \mid$ is the number of edges present in a subgraph $\acute{G}$ and $\mid \acute{E} \mid_{max}$ is the maximum possible number of edges in same subgraph $\acute{G}$. Except density, They have also used *Clustering Property* which is as follows-

$$cp_{nk} = \frac{\mid E_{nk} \mid}{d_k \times \mid N_k \mid} \tag{2.4}$$

Here, $\mid E_{nk} \mid$ is the total number of edges between the node n and each of the nodes of cluster k; and N_k is the total number of vertices of cluster k. Using *Clustering property* and *density* of each node with its neighbors forms clusters. This process continues until it reaches density threshold. The algorithm is known as DPClus. The time complexity of the algorithm is $O(n^2)$. This algorithm cannot solve the second and common shortcoming of Li et al. [20]. As well as its computational time is very high for larger protein interaction networks.

2.1.4 Pei et al. 2007

Pei et al [25] has designed a new algorithm based on density of the network. Their algorithm is faster than Altaf et al [1]'s algorithm. They have also solved the first problem of Li et al [20]. To do this, they have modified the equation of density which is as follows-

$$den(\acute{G}) = \frac{\sum v \varepsilon \acute{V} \frac{|N_v \cap \acute{V}}{|\acute{V}-1|}}{|\acute{V}|} \tag{2.5}$$

where N_v is the neighbour list of vertex v and $\acute{V}$ is the list of the vertices of subgraph $\acute{G}$. As well as they have introduced a new measure called *SIGnificance BOUNDary subgraph quality*, denoted as $Q_{SigBound}(\acute{G})$. It calculates the boundary significance of a subgraph $\acute{G}$ with its neighbor subgraphs. In the algorithm, at first a seed edge is selected by using the definition of *center* of a graph. In the second step, it selects a seed vertex among the connected vertices of seed edge. After selecting seed vertex and edge, it calculates the density and $Q_{SigBound}(\acute{G})$ for each vertex and edge to form cluster with seed edge or vertex. This process continues till the density and $Q_{SigBound}(\acute{G})$ of a cluster increase. After that, the cluster is separated from the network. The algorithm starts again for finding the rest of the clusters. This algorithm is known as *Seed-Refine algorithm*. The time complexity of the algorithm is $O(n\bar{C})$ where $\bar{C}$ is the average size of clusters of a PIN. Though this algorithm solves the first shortcomings of Li et al's [20] algorithm, it has three major problems-

- Seed edge has two vertices. There is a possibility to select wrong vertex as seed vertex which cause improper clustering
- It cannot solve the second problem of Li et al [20].
- It cannot work on PINs which have parallel edges and self loops.

2.1.5 Summary

Here we have shown the summery of the density based methods designed for clustering protein interaction networks in Table- 2.1. In the Table- 2.1, we have arrayed major contribution, worst case time complexity and major shortcomings of each algorithms of density based method.

Table 2.1: Summary of previous algorithms based on density based method

Algorithm	**Major Contribution**	**Worst case time complexity**	**Major problem**
Seed Refine [25]	Introduces and uses new definition of density and $Q_{SigBound}$	$O(n\bar{C})$	Cannot work on PINs having parallel edges and self loops and cannot classify vertices of degree one with their neighbors and time consuming method
DPClus [1]	Uses newly defined density definition and clustering property to cluster PINs	$O(n^2)$	Cannot classify vertices of degree one with their neighbors and time consuming method
LCMA [20]	Introduces and uses clustering coefficient to identify the cliques and Network affinity to predict the densed subgraph	$O(lk^2v)$	Cannot classify vertices of degree one with their neighbors
Spirin et al [29]	Combines the concepts of MCL and SPC algorithms to find the clusters	$O(n^2k^2)$	Not time efficient

2.2 Graph partitioning approaches

In the current Section, we discuss the graph partitioning approaches to predict the communities in PINs. All partitioning algorithms detect the edges which are acted as the bridge between communities. By removing bridge edges, the algorithms identify the clusters of PINs.

2.2.1 Dongen 2000

S. V. Dongen designed an algorithm based on graph partition in his Ph.D. thesis. This algorithm is known as *Markov Cluatering* algorithm, in short *MCL*. MCL algorithm was designed based on random walk between the nodes of the graph. Random walk is calculated by exponential normalized adjacency matrix and inflation parameter r. After calculating random walk, MCL removes the edges with lower random walk values to separate the clusters from network. This algorithm is commonly used in graph clustering. The worst case time complexity of the MCL algorithm is $O(n^2p)$ where n and p are the number of nodes in PINs and passes or random walk respectively. Its efficiency depends on the selection of inflation parameter r and power parameter e. Wrong selection of r and e makes the algorithm inefficient.

2.2.2 King et al. 2004

In the year 2004, a cost function based community detection algorithm was designed by King et al [17] for predicting protein complexes. Their algorithm is known as *Restricted Neighborhood Search Clustering (RNSC)* algorithm. This algorithm is devised on basically Tabu search meta-heuristics. RNSC algorithm searches the space of the partition (bridge edges) and assign a cost by using cost function which is not clearly mentioned in their paper. After that, the algorithm separates the clusters from others by removing low cost edges. RNSC gives good results for Giot et al [11]'s fruit fly's protein interaction network. Except this species, RNSC algorithm finds fewer complexes for all species. Besides, the result of the algorithm heavily depends on the initial value which is random.

2.2.3 Graph Entropy Algorithm

Recently, Kenley et al [16] has designed a new graph partition algorithm based on graph entropy. The graph entropy is defined based on the probability distribution of its inner links and outer links. It is denoted as $e(G)$.

$$e(G) = \sum v \varepsilon V e(v) \tag{2.6}$$

where

$$e(v) = -p_i(v)\log_2 p_i(v) - p_o(v)\log_2 p_o(v) \tag{2.7}$$

Here $p_o(v)$ and $p_i(v)$ denotes the probability of v having outer link and inter links respectively. The graph entropy measure the cluster quality effectively. A graph with lower entropy indicates that the vertices in the cluster have more inner links and less outer links.

The algorithm starts it working by selecting a random seed vertex and its neighbors as seed cluster. After that, it iteratively adds or delete the vertices on the border of the cluster to minimize the graph entropy. To produce a final set of cluster, the process of seed selection and optimal cluster generation is repeatedly performed until no seed vertex is remaining. This algorithm is known as *Graph Entropy* algorithm. The time complexity of the algorithm is $O(n^2)$. Though it is time consuming algorithm, it can efficiently find the complexes and modules.

2.2.4 Summary

Here we have shown the summery of the graph partition based methods designed for clustering protein interaction networks in Table- 2.2. In the Table- 2.2, we have arrayed major contribution, worst case time complexity and major shortcomings of each algorithms of graph partition based method.

Table 2.2: Summary of previous algorithms based on graph partitioning method

Algorithm	**Major Contribution**	**Worst case time complexity**	**Major problem**
Graph Entropy [16]	Introduces and uses graph entropy concept	$O(n^2)$	Time consuming approach
RNSC [17]	Introduces and uses tabu search meta heuristic and cost functions	Not mentioned	Except fruit fly, it can detect very few clusters or communities accurately
MCL [8]	Uses random walk approach to find the clusters	$O(n^2 p)$	Dependency on the parameters r and e

2.3 Hierarchical based methods

In this Section, we have discussed the previous hierarchical methods which are used to identify the communities in PINs. Hierarchical methods use some measures to calculated the similarity or compactness between nodes or communities for forming clusters. These methods can be classified into two groups: *agglomerative* and *divisive*.

In agglomerative approach, at first all vertices are considered as individual clusters which are known as *singletons*. After calculating similarity or compactness by using several measures or metrics (edge betweenness, edge clustering coefficient, edge clustering value etc.), this approach merges two communities according to their most similarity or compactness. This process continues until the graph remains one community.

Divisive approach is opposite of agglomerative approach. In this approach, a network is considered as a community. After calculating similarity or compactness by using measure, it divides the community into multiple communities according to their less similarity or compactness. It continues until all vertices are represented as singletons.

2.3.1 Ravasz et al. 2002

Ravasz et al [27] proposed an agglomerative hierarchical clustering approach to identify the protein complexes from PINs. Their algorithm has been designed on the basis of density of the clusters. For identifying or calculating the density of the subgraph, they have used *Clustering Coefficient Values* which is as followed-

$$cc(\acute{G}) = \frac{2 \times e}{\mid k_i \mid \times (\mid k_i \mid -1)} \tag{2.8}$$

where e is the number of edges connecting the k_i nearest neighbors of node i. They are first designers for introducing hierarchical methods for clustering protein interaction networks. The time complexity of the algorithm is $O(n^2)$. The problem of this method- it cannot work properly on scale free PINs. It takes more time to execute the algorithm for larger PINs having more than hundred thousands proteins.

2.3.2 Girvan et al. 2002

Girvan and Newman proposed a new approach in hierarchical methods to classify PINs into multiple clusters in their paper [12]. This algorithm is known as *GN* algorithm. In this algorithm, Girvan and Newman introduced a new measure for clustering PINs. This measure is known as *edge betweenness*. The *edge betweenness* of an edge is the number of shortest paths between pairs of nodes that run along it. If there is more than one shortest path between a pair of nodes, each path is assigned equal weight such that the total weight of all of the paths is equal to unity. If a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness (at least one of them). By removing these edges, the groups are separated from one another and so the underlying community structure of the network is revealed. This algorithm follows *divisive* approach. The time complexity of GN algorithm is $O(m^2n)$ where m and n are the number of edges and vertices respectively. GN algorithm is one of the most used algorithm in the field of bioinformatics. Though it is most commonly used, GN algorithm has two major shortcomings-

- It is very time consuming algorithm. Generally, the number of edges in PINs are larger than the number of vertices.
- It also faces the problem of clustering vertices of degree one.

2.3.3 Newman 2003

Newman designed a new hierarchical algorithm which is faster than GN algorithm. He proposed his algorithm in his paper ”Fast algorithm for detecting community structure in network” [22]. This agglomerative algorithm was designed on the basis of *modularity, Q-*

$$Q = \sum_{i=1}^{k} (e_{ii} - a_i^2), \tag{2.9}$$

where, where e_{ii} is the fraction of edges with both end vertices in the same community i, and a_i is the fraction of edges with at least one end vertex in community i. In agglomerative steps, two communities are merged together if ΔQ increases or fixed. ΔQ follows following equation-

$$\Delta Q = 2(e_{ij} - a_i * a_j) \tag{2.10}$$

The time complexity of the Newman's algorithm is $O((m+n)n)$ for PINs. Though Newman's algorithm is faster in comparison with GN algorithm, for PINs still it is time consuming algorithm. On the other hand, for rat and mouse PINs, it suffers the second problem of GN algorithm.

2.3.4 Radicchi et al 2004

Radicchi et al. [26] designed a new hierarchical algorithm based on the definition of weak and strong communities. According to the definitions of weak and strong modules, they introduced edge clustering coefficient in hierarchical algorithm to improve time complexity. After calculating edge clustering value of each edge, their algorithm works like GN algorithm but at every step, the removed edges are those with the smallest value of $\tilde{C_{i,j}}$.

$$\tilde{C_{i,j}} = \frac{Z_{i,j}^{(3)}}{min[(k_i - 1), (k_j - 1)]} \tag{2.11}$$

Here, $\tilde{C_{i,j}}$ is the modified edge clustering coefficient, $Z_{i,j}^{(3)}$ is the number of triangles built on that edge (i, j) and $min[k_i - 1, k_j - 1]$ is the maximal possible number of them. The time complexity of the algorithm is $O(m^2)$. Though the algorithm is faster than GN algorithm, still it is slow method for large PINs. On the other hand, it has another problem-if a community does not have any triangle or having a cycle, this algorithm is not efficient enough [30].

2.3.5 Clauset et al 2004

Dr. Aaron Clauset and his supervisors Dr. M.E.J. Newman and Dr. C. Moore has designed a new agglomerative hierarchical algorithm which used the concept of ΔQ. Their algorithm is the improved version of Newman's ΔQ hierarchical algorithm [22]. For improving the performance, they modified ΔQ-

$$\Delta Q_{i,j} = \begin{cases} \frac{1}{2m} - \frac{d_i * d_j}{(2m)^2} & \text{if } i \text{ and } i \text{ are connected} \\ 0 & \text{otherwise.} \end{cases} \tag{2.12}$$

Where d_i is the degree of vertex i. For improving time complexity, they used *max heap* and *sparse matrix* to store the value of ΔQ. Their proposed algorithm is known as *CNM* algorithm. They algorithm works as follows-

- It calculates the initial values of $\Delta Q_{i,j}$ and store the largest value of each row of the sparse matrix in max heap.
- It selects the largest $\Delta Q_{i,j}$, merge two communities i and j.
- After merging, it recalculates the $\Delta Q_{i,j}$ and repeats the above step until one community remains.

The time complexity of the algorithm is $O(mh \log_2 n)$ where h is the depth of dendrogram. This algorithm improves the time complexity in significant amount. But it suffers the common problem in hierarchical approach- clustering vertices of degree one in separate community from their neighbours.

2.3.6 Luo et al 2007

In 2007, a new agglomerative approach was introduced based on *edge betweenness* by Luo et al [21]. Like GN algorithm, it calculates the edge-betweenness of all edges and sorted them in ascending order. After that it follows-

- If the edge connects the vertices in same subgraph, it is added to the subgraph [21].
- If the edge connects the vertices in two different subgraph, two subgraphs are added with satisfying one of two conditions-
  - Two subgraphs are non-modules (non-module subgraph contains only one vertex).
  - One subgraph is module and another one is non-module.

The process continues till $\mid E \mid \neq 0$. This algorithm is known as *MoNet* algorithm. The time complexity of MoNet algorithm is $O(m^2n)$. This algorithm also suffers same the shortcomings as the GN algorithm.

2.3.7 Li et al. 2008

Min Li and her colleagues designed an agglomerative hierarchical algorithm based on edge clustering value [19]. They have modified the formula of edge clustering coefficient. Their modified edge clustering coefficient is as follows:

$$CC = \frac{\mid N_i \cap N_j \mid + 1}{min(d_i, d_j)} \tag{2.13}$$

where, N_i and d_i are the neighbor list and degree of vertex i respectively. As per their algorithm, edge clustering coefficient values are calculated and sorted in descending order for all edges. After that, all singletons are merged together by the edges which are sorted in descending order according to edge clustering coefficient values. The algorithm is known as FAG-EC algorithm. The time complexity of the algorithm is $O(\bar{d}^2 m)$. Here $\bar{d}$ is the average degree of the network. The algorithm has one major problem. If in a PIN, a good cluster has no triangle or has cycle, the edge clustering coefficient formula produces low values for its (cluster's inside) edges. For this reason, the FAG-EC algorithm cannot produce accurate clusters.

2.3.8 Wang et al 2011

Min Li and her colleagues proposed a new algorithm based on the problem of their previous algorithm *FAG-EC* which is known as *HC-PIN* algorithm [30]. In their algorithm, they introduced a new measure to overcome the problem of FAG-EC algorithm. This measure is called *Edge Clustering Value (ECV).*

$$ECV = \frac{\mid N_u \cap N_v \mid^2}{\mid N_u \mid \times \mid N_v \mid} \tag{2.14}$$

Where N_u is the set of neighbors of vertex v. They also redesigned ECV for weighted PINs which is as follows-

$$ECV^* = \frac{\sum_{k \varepsilon I_{u,v}} w(u,k) \times \sum_{k \varepsilon I_{u,v}} w(v,k)}{\sum_{s \varepsilon N_u} w(u,s) \times \sum_{s \varepsilon N_u} w(u,s)} \tag{2.15}$$

where $I_{u,v}$ denotes the set of common vertices N_u and N_v. The Equation 2.14 is the special case of Equation 2.15. Except the calculation of ECV, HC-PIN algorithm works like FAG-EC algorithm. The time complexity of HC-PIN algorithm is $O(\bar{d}^2 m)$. Though the HC-PIN algorithm solves the problem of FAG-EC algorithm and its working capability over weighted PINs, it suffers a common problem of hierarchical approach: classifying the vertices of degree one in separate clusters from their neighbors.

2.3.9 Summary

Here we have shown the summery of the hierarchical methods designed for clustering protein interaction networks in Table- 2.3. In the Table- 2.3, we have arrayed major contribution, worst case time complexity and major shortcomings of each algorithms of hierarchical method.

Table 2.3: Summary of previous algorithms based on hierarchical method

Algorithm	**Major Contribution**	**Time complexity**	**Major problem**
HC-PIN [30]	Solves the problem of using edge clustering coefficient	$O(\bar{d}^2 m)$	Cannot classify vertices of degree one with their neighbors
FAG-EC [19]	Improves the time complexity	$O(\bar{d}^2 m)$	Cannot classify vertices of degree one with their neighbors and cannot identify the cluster with no triangle or having cycle
MoNet [21]	Uses edge betweenness in agglomerative approach	$O(m^2 n)$	Execution time and cannot classify vertices of degree one with their neighbors
CNM [7]	Improves the time complexity by modifying ΔQ, sparse matrix and Max heap	$O(mh \log_2 n)$	Cannot classify vertices of degree one with their neighbors
Radicchi et al [26]	Defines strong and weak community, uses edge clustering coefficient in divisive approach	$O(m^2)$	Cannot classify vertices of degree one with their neighbors and cannot identify the cluster with no triangle or having cycle
Newman [22]	Introduces ΔQ to improves time complexity	$O((m+n)n)$	Cannot classify vertices of degree one with their neighbors.
GN [12]	Introducing edge concept in Divisive hierarchical approach and uses edge betweenness	$O(m^2 n)$	Cannot classify vertices of degree one with their neighbors.
Ravasz et al [27]	First hierarchical algorithm for clustering PINs	$O(n^2)$	Cannot classify vertices of degree one with their neighbors and cannot identify the cluster with no triangle or having cycle.

Chapter 3

Fast Agglomerative Clustering Algorithms

We have discussed our proposed algorithm: *FAC-PIN* and premetric: *relative vertex-to-vertex clustering value* in detail in this Chapter. Our designed FAC-PIN algorithm has been developed based on the second, third and fourth properties of PINs. We have briefly described the premetric *relative vertex-to-vertex clustering value* in Section 3.1 and FAC-PIN algorithm with its computational complexity in Section 3.2.

3.1 Relative Vertex-to-Vertex Clustering Value

The edge clustering value, $ECV(u,v)$, used in HC-PIN [30], is a similarity metric between the two vertices u and v of an edge (u,v) and which, roughly speaking, tells how likely u and v lie in the same module (i.e., cluster). This is also true with the edge clustering coefficient, $C^{(3)}_{u,v}$, of [26]. However, in complex networks following the power law (i.e., scale-free networks), it is reasonable to assume that the likelihood of a vertex u to lie in the same module as v (or, to lie in the module containing v), is not equal to the likelihood of v to lie in the module containing u. This assumption stems from the principle of *preferential attachment* in scale-free networks which states that a new node u is likely to *attach* to a high-degree node v than to a low degree node. This is not reciprocal, and hence, clearly suggesting that the likelihood is not symmetric and that it is larger for u to be in a cluster with v than for v to be in cluster with u (if we assume that v is a high-degree node). The similarity metrics $ECV(u,v)$ and $C^{(3)}_{u,v}$ treat equally both endpoints of edges (u,v) irrespective of their degrees. Also, another issue is that both $ECV(u,v)$ and $C^{(3)}_{u,v}$ require vertices u and v be connected by an edge. This requirement is quite restrictive and we aim to extend to the case in which pair (u,v) is not an edge while still being able to decide if both vertices are in the same cluster. Finally, as stated earlier in previous section, current hierarchical approaches have the common problem of classifying low-degree vertices (peripheral to dense subnetwork modules) into separate clusters rather than merging them with their neighboring modules. In the following paragraph, we present a new measure which aims to address these issues.

Let N_u be the set of neighbors of vertex u in an undirected graph $G=(V,E)$. We define $N_u^+ = N_u \cup \{u\}$ as the neighbor set of u augmented with u itself. Given two vertices u and v, we define the clustering value of u relative to v as:

$$R(u \dashrightarrow v) = \frac{|N_u^+ \cap N_v^+|}{|N_u^+|} \tag{3.1}$$

$R(u \dashrightarrow v)$ is a premetric that ranges from 0 to 1; that is, it is a measure which does not satisfy the axiom of symmetry and the triangle inequality but satisfies the axioms of self-similarity and minimality. A vertex u with a larger clustering value given another vertex v is more likely to lie in the cluster containing v. In the following $C(a)$ denotes the cluster containing a given vertex a, and we assume that $C(a)$ satisfies the *weak sense* definition of a community [26] (we use the term ws-cluster, hereafter). The following describe the properties of $R(u \dashrightarrow v)$.

Given an edge (u,v), $R(u \dashrightarrow v)$ is maximal (i.e. equals 1) if and only if $|N_u^+| = |N_u^+ \cap N_v^+|$. There are two cases achieving the maximum given edge (u,v):

1. when u has degree one
2. when both u and v have the same degree and $|N_u^+| = |N_v^+|$ that is, they have the same neighbors.

In either case, If sub-network $C(v)$ (respectively, the induced sub-network of G for subset N_v^+) is a ws-cluster then $\{u\} \cup C(v)$ (respectively, $\{u\} \cup N_v^+$) is a also a ws-cluster.

Given an edge (u,v), $R(u \dashrightarrow v)$ is minimal when u is the highest degree vertex in G and v has degree 1; that is, $R(u \dashrightarrow v) = \frac{2}{1+deg(u,G)}$ and $deg(u,G)$ is maximal. In such case, $R(v \dashrightarrow u)$ is maximal (i.e. equals 1), and hence, $C(u) \cup \{v\}$ (respectively, $N_u^+ \cup \{v\}$) is a ws-cluster if $C(u)$ (respectively, N_u^+) is a ws-cluster. Here, we have found that $R(v \dashrightarrow u)$ solves the problem of clustering the vertices of degree one.

Given an edge (u,v), assume the degrees of vertices u and v in G are such that $deg(u,G) = deg(v,G) = d$ is maximal and that u and v do not share any other neighbors. Then, we have $R(u \dashrightarrow v) = R(v \dashrightarrow u) = \frac{2}{1+d} \leq 0.5$ assuming $d \geq 3$. In this case, $\{u\} \cup C(v)$ (or N_v^+) is not a ws-cluster, and, $\{v\} \cup C(u)$ (or N_u^+) is not a ws-cluster. Consider the induced subgraph of G on $N_u^+ \cup N_v^+$, we define the *local betweenness value* of edge (u,v) as the percentage of paths from vertices in $N_u \setminus N_v$ to vertices in $N_v \setminus N_u$ going through edge (u,v). Given the number of common neighbors between u and v, $|N_u \cap N_v|$, the local betweenness of edge (u,v) is thus $l(u,v) = 100 \cdot \frac{1}{|N_u \cap N_v|+1}$. Given two connected high-degree vertices u and v, the local edge betweenness value $l(u,v)$ increases when $|N_u \cap N_v|$ decreases, and hence, it corresponds to when both $R(u \dashrightarrow v)$ and $R(v \dashrightarrow u)$ values are small at the same time. Edges with high local betweenness values are edges connecting two clusters, and therefore, vertices u and v should not lie in the same cluster.

Finally, our relative vertex clustering values implements the ideas behind the edge clustering coefficient, $C^{(k)}_{u,v}$, of [26], since for a given vertex v and a neighbor u the number of triangles given edge (u,v) is exactly $|N_u \cap N_v|$; and u will be included into $C(v)$ whenever most of the neighbors of u (excluding v) are in $N_u \cap N_v$. This is also true even when (u,v) is not an edge; in such case, $|N_u \cap N_v|$ relates to the number of squares containing vertices u and v. On the other hand, we break through the limitations of [26] as in the edge clustering value, $ECV(u,v)$ of [30], by not assuming the existence of closed loops in a network, such as triangles or high-order loops. The relative vertex clustering value $R(u \dashrightarrow v)$ also improves $ECV(u,v)$ since neighbors u of v which have most of their neighbors forming a triangle with v are selected for inclusion in $C(v)$. Searching for vertices u which form a cluster with v is also more efficient than searching for edges (u,v) that makes a cluster since the number of edges is larger than the number of vertices in dense subgraphs.

In summary, the values $R(u \dashrightarrow v)$ and $R(v \dashrightarrow u)$ for edge (u,v) can be used as a quick test for deciding whether u (respectively, v) should be merged with the cluster $C(v)$ (respectively, $C(u)$) such that $\{u\} \cup C(v)$ (respectively, $\{v\} \cup C(u)$) remains a ws-cluster.

We have designed $R(u \dashrightarrow v)$ based on the concept of preferential attachment property of protein interaction networks. It calculates the likelihood value of any vertex v to form cluster with another vertex u. On the other hand, *Edge Clustering Value* of Equation 2.14 was designed for each edge not for vertices. Our designed $R(u \dashrightarrow v)$ is a premetric [1]. But *ECV* of HC-PIN algorithm is similarity metric.

[1] A metric satisfies axioms of positivity and positive definiteness

3.2 The FAC-PIN Algorithm

3.2.1 The algorithm

Our proposed fast agglomerative clustering algorithm for protein interaction networks, FAC-PIN in Algorithm 1, goes as follows. Given a PIN $G = (V, E)$, we initially consider each vertex as a singleton, and sort the vertices $v \in V$ in descending order of their degrees $deg(v, G)$ in G. Here we have used the second property of community structure. After sorting, in an iterative manner, we select the next highest-degree vertex v from the sorted list, and compute the values $R(u \dashrightarrow v)$ and $R(v \dashrightarrow u)$ for each neighbor u of v, and then decide depending on these two values and a threshold $\alpha, 0 \leq \alpha \leq 1$, whether u should be included in $C(v)$ or not.

In the FAC-PIN algorithm, a neighbor u of vertex v is added to the current $C(v)$ when the majority of the neighbors of u are in $N_u^+ \cap N_v^+$, that is when:

1. $R(u \dashrightarrow v) = 1$, in which case either u has degree 1, or u and v have the same degree and the same set of neighbors;

2. $R(u \dashrightarrow v) > R(v \dashrightarrow u) > \alpha$, in which case u have smaller degree than v and most of the neighbors of u are in the intersection; and

3. $R(u \dashrightarrow v) = R(v \dashrightarrow u)$ and the size of the intersection is larger than the total set of neighbors of u and v which are not in the intersection.

3.2.2 Computational Complexity

Let $n = |V|$ be the number vertices, $m = |E|$ be the number of edges, and $\bar{d}$ be the average degree of all vertices, that is $\bar{d} = \frac{1}{n} \sum_{v \in V} deg(v, G)$. The complexity of sorting the vertices by

Algorithm 1 The *FAC-PIN* Algorithm

```
Input: $G = (V,E)$: undirected PIN graph
       $\alpha$: threshold parameter
Output: $P_k = \{C_1,\dots,C_k\}$: identified collection of modules
  {Initialization phase}
  for every $v_i \in V$ do
    $C(v_i) \leftarrow \{ \{v_i\}, \emptyset \}$; {each vertex is a singleton cluster}
  end for
  Sort all vertices to a priority-queue $H$ in non-increasing order of their degrees;
  {Community detection phase}
  repeat
    $v \leftarrow H$; {select next highest-degree vertex in $H$}
    for all $u \in N_v$ not yet merged into a cluster do
      if $[R(u \dashrightarrow v) = 1]$ Or $[R(u \dashrightarrow v) > R(v \dashrightarrow u) > \alpha]$ then
        $C(v) \leftarrow C(v) \cup \{ \{u\}, \{(u,v)\} \}$;
        $C(u) \leftarrow C(v)$;
      else
        if $[R(u \dashrightarrow v) = R(v \dashrightarrow u)] \geq \alpha$ And $[deg(u,G) + deg(v,G) - 1 \leq |N_u \cap N_v|]$
        then
          $C(v) \leftarrow C(v) \cup \{ \{u\}, \{(u,v)\} \}$;
          $C(u) \leftarrow C(v)$;
        end if
      end if
    end for
  until $H = \emptyset$
  $U \leftarrow V$;
  $i \leftarrow 1$;
  {Compute the partition $P_k$}
  while $U \neq \emptyset$ do
    $v \leftarrow$ randomly select a vertex from $U$;
    $C_i \leftarrow C(v)$;
    $U \leftarrow U \setminus \{u \mid C(u) = C(v)\}$;
    $i \leftarrow i + 1$;
  end while
  return $P_k \leftarrow \{C_1,\dots,C_k\}$;
  Evaluate modularity $Q(P_k)$ of partition $P_k = \{C_1,\dots,C_k\}$;
```

their degree is $O(n)$ by using the *counting sort* method, and the complexity of computing the partition after the community detection phase is also $O(n)$. Let the maximum node degree in G be $d_{\max} = \max_{v \in V} deg(v, G)$. The complexity of computing $R(u \dashrightarrow v)$ given vertices u and v in the "for-loop" of FAC-PIN is $O(d_{\max})$. The complexity of the "for-loop" is then $O(d_{\max}^2)$, and hence, the total complexity of the "repeat-loop" (and thus of FAC-PIN) is $O(nd_{\max}^2) \ll O(n^3)$. Since PINs are power-law networks then the majority of the proteins interact with only very few proteins, and thus the average degree $\bar{d}$ is generally small and can be considered a constant [30]; that is, we can use $\bar{d}$ as the principal variable for measuring the complexity of community detection methods. As such, then the worst case time complexity of FAC-PIN is $O(n\bar{d}^2) \ll O(nd_{\max}^2) \ll O(n^3)$. The worst case time complexity of the HC-PIN algorithm of [30] is $O(m\bar{d}^2)$ and is larger than that of FAC-PIN since $n \lll m$ in PINs. We note that HC-PIN is currently the fastest hierarchical method described in literature for clustering PINs, as far as we know.

Chapter 4

Computation experiments, results and discussions

We tested our FAC-PIN algorithm using several test procedures to understand its working capability on PINs. For experiment purpose we used protein interaction networks of sixteen different species. We set the threshold parameter α in the FAC-PIN algorithm to values 0.75, 0.5, 0.25, 0.125, 0.0625, 0.03125, 0.0156, 0.0078 and 0.0039 and ran FAC-PIN with each of these values. After generating clusters of each protein interaction network for each α, a mathematical function is used to evaluate the clusters of a protein interaction network. This mathematical function is known as scoring function. Scoring function defines the compactness or density of a cluster. Most of the cases, higher scoring function indicates clusters of the PIN are dense and loosely connected with rest of the clusters. In the experiments, modularity Q [7, 22] and w-log-v [18] are used as scoring functions. In this case, the clustered protein interaction network of highest scoring function is considered as clustered PIN of a specific species. After clustering, clusters of PIN compared with the physical complexes and identified functional modules of PIN of a species by using complex

Table 4.1: Dataset used in experiments

Species	Scientific name	Proteins	Interactions	Used symbol	Database
Baker's yeast	*Saccharomyces cerevisiae*	4572	49830	Y_{4K}	2
		5697	50675	Y_{5K}	3
Cattle	*Bos taurus*	5737	113888	C_{5K}	3
Dog	*Canis lupus familiaris*	2932	38647	D_{2K}	3
E. coli	*Escherichia coli*	2817	13841	E_{2K}	1
Finch bird	*Taeniopygia Guttata*	3929	74314	FB_{3K}	3
Flowering plant (Thale cress)	*Arabidopsis thaliana*	2651	5236	FP_{2K}	1
Fruit fly	*Drosophila melanogaster*	8366	25611	FF_{8K}	1
Frog	*Xenopus Tropicalis*	5473	122706	FG_{5K}	3
Jungle fowl (Chicken)	*Gallus gallus*	4960	112250	J_{4K}	3
Human	*Homo sapiens*	12994	135935	H_{12K}	3
		8997	34935	H_{8K}	1
Mouse	*Mus musculus*	2888	4372	M_{2K}	1
Rat	*Rattus norvegicus*	1148	1307	RA_{1K}	1
Rice	*Oryza sativa*	3778	320570	RI_{3K}	3
Round worm	*Caenorhabditis elegans*	4303	7747	RW_{4K}	1
Wild boar	*Sus Scrofa*	5303	119920	W_{5K}	3
Zebra fish	*Danio rerio*	8188	274358	Z_{8K}	3

validation and functional module testing process respectively. we also compared the results of FAC-PIN algorithm with *HC-PIN* and *CNM* algorithms. Whole experiment process and results are discussed in different sections of current Chapter.

4.1 Datasets

The PINs of sixteen different species were obtained from the PINALOG site[1], the BioGRID database[2] and REACTOME database[3]. The sixteen species are listed along with their number of proteins and interactions in Table- 4.1.

From the Table- 4.1, it is found that the number of edges (interactions) is quite larger than the number of vertices (proteins).

[1]http://www.sbg.bio.ic.ac.uk/~pinalog/downloads.html

[2]thebiogrid.org

[3]http://www.reactome.org/download/all_interactions.html

4.2 Testing by Scoring Functions

For given a clustering result (i.e. a partition) $P_k = \{C_1, \ldots, C_k\}$ with k clusters, we used the popular modularity function Q as one of the scoring function, introduced by Newman and Girvan [7].

$$Q = \sum_{i=1}^{k} (e_{ii} - a_i^2), \tag{4.1}$$

where e_{ii} is the fraction of edges with both end vertices in the same community i, and a_i is the fraction of edges with at least one end vertex in community i. k is the number of clusters of the PIN. Larger values of Q correspond to more distinct community structures in PINs. It means *in-degree* of a community i is larger than the *out-degree* of the same community i. Though Q is widely used, it is known to have serious limitations which has been discussed at length in [9]. The second partition scoring function we used has been introduced in [18] and is defined as

$$w\text{-}\log\text{-}v = \sum_{i=1}^{k} (e_{ii} \times \log a_i). \tag{4.2}$$

Function w-$\log$-v allows for more diverse cluster sizes than function Q, It produces negative values. Its smaller values corresponds to better clustered structures.

As said above, we ran FAC-PIN many times (each with different values of threshold parameter α), then evaluate the clustered structure of the communities obtained by FAC-PIN, and then retain the results giving the best scoring values. We also implemented the best-performing HC-PIN algorithm of [30] and the hierarchical method of [7] which we denote as the CNM algorithm. The HC-PIN and CNM methods were run on the same PIN data as the FAC-PIN approach. For HC-PIN, we set the two parameters λ and s as in [30]

(CNM has no parameters). The results of the three methods are given in Tables 4.2, 4.3, 4.4, 4.5, 4.6 and 4.7, respectively in terms of scoring values Q and w-log-v, and running times obtained by each method for each species. As well as we have showed the histogram of modularity Q, w-log-v and log-log plot of time comparison of *FAC-PIN*, *HC-PIN* and *CNM* algorithms in Figures- 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6 respectively.

Table 4.2: Q results of *FAC-PIN*, *CNM* and *HC-PIN* for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog.

Algorithms	Y_{4K}	Y_{5K}	C_{5K}	D_{2K}	E_{2K}	FB_{3K}	FP_{2K}	FF_{8K}	FG_{5K}
FAC-PIN	0.5415	0.5110	0.7288	0.7566	0.1492	0.7874	0.9422	0.6486	0.7432
CNM	0.5391	0.1412	0.6969	0.6850	0.0587	0.7199	0.7861	0.3116	0.6909
HC-PIN	0.5401	0.0387	0.5265	0.6405	0.0023	0.6075	0.7819	0.0086	0.4907

Table 4.3: Q results of *FAC-PIN*, *CNM* and *HC-PIN* for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish.

Algorithms	H_{8K}	H_{12K}	J_{4K}	M_{2K}	RA_{1K}	RI_{3K}	RW_{4K}	W_{5K}	Z_{8K}
FAC-PIN	0.5893	0.7827	0.7540	0.7644	0.7897	0.5401	0.7484	0.7536	0.7692
CNM	0.4768	0.2858	0.7000	0.4781	0.5457	0.5215	0.4057	0.7040	0.2294
HC-PIN	0.2768	0.0126	0.6527	0.5015	0.4502	0.1791	0.2928	0.5180	0.7527

Table 4.4: $-(w\text{-}\log\text{-})v$ results of *FAC-PIN*, *CNM* and *HC-PIN* for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog.

Algorithms	Y_{4K}	Y_{5K}	C_{5K}	D_{2K}	E_{2K}	FB_{3K}	FP_{2K}	FF_{8K}	FG_{5K}
FAC-PIN	1.301	0.521	1.141	1.630	0.262	1.359	3.603	1.517	1.208
CNM	1.299	0.481	0.7173	1.278	0.192	1.119	2.866	1.233	0.997
HC-PIN	1.299	0.028	0.3162	0.998	0.019	0.847	3.071	0.072	0.516

Table 4.5: $-(w\text{-}\log\text{-}v)$ results of *FAC-PIN*, *CNM* and *HC-PIN* for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish.

Algorithms	H_{8K}	H_{12K}	J_{4K}	M_{2K}	RA_{1K}	RI_{3K}	RW_{4K}	W_{5K}	Z_{8K}
FAC-PIN	1.366	1.941	1.568	2.634	2.525	1.615	2.094	1.048	0.773
CNM	1.197	1.269	1.488	1.530	1.699	1.585	1.819	1.283	0.756
HC-PIN	0.566	0.113	1.283	1.805	1.558	0.237	1.809	0.760	0.262

As we see in both Tables 4.2, 4.3, 4.4 and 4.5, FAC-PIN outperformed both the HC-PIN and CNM methods in all given PINs. We note that, as the size of the PINs increases, in

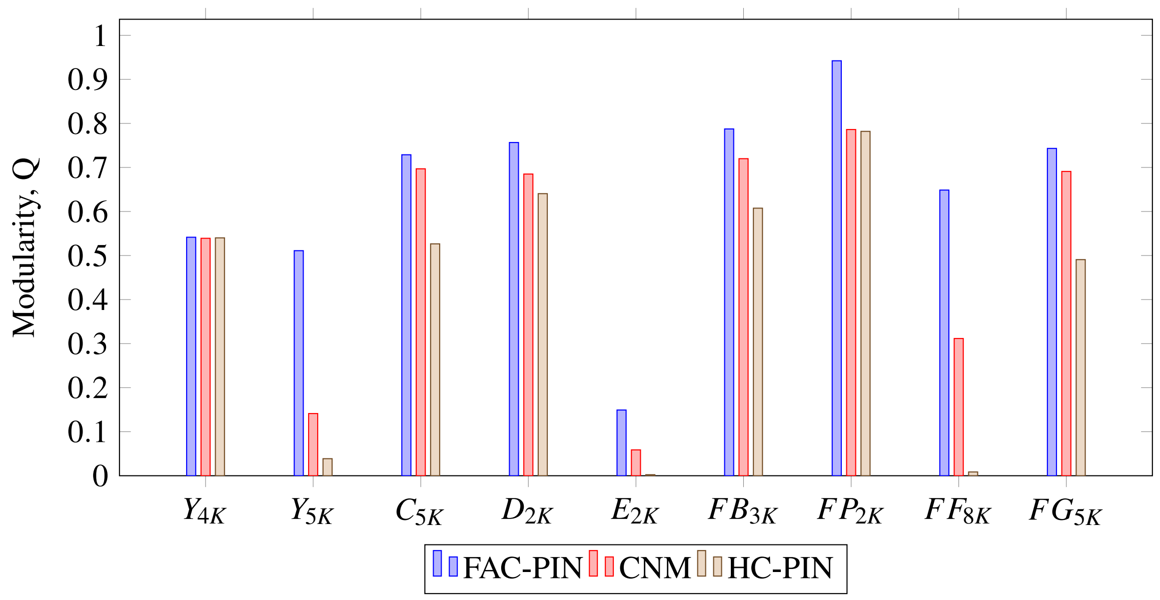

Figure 4.1: Modularity comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog

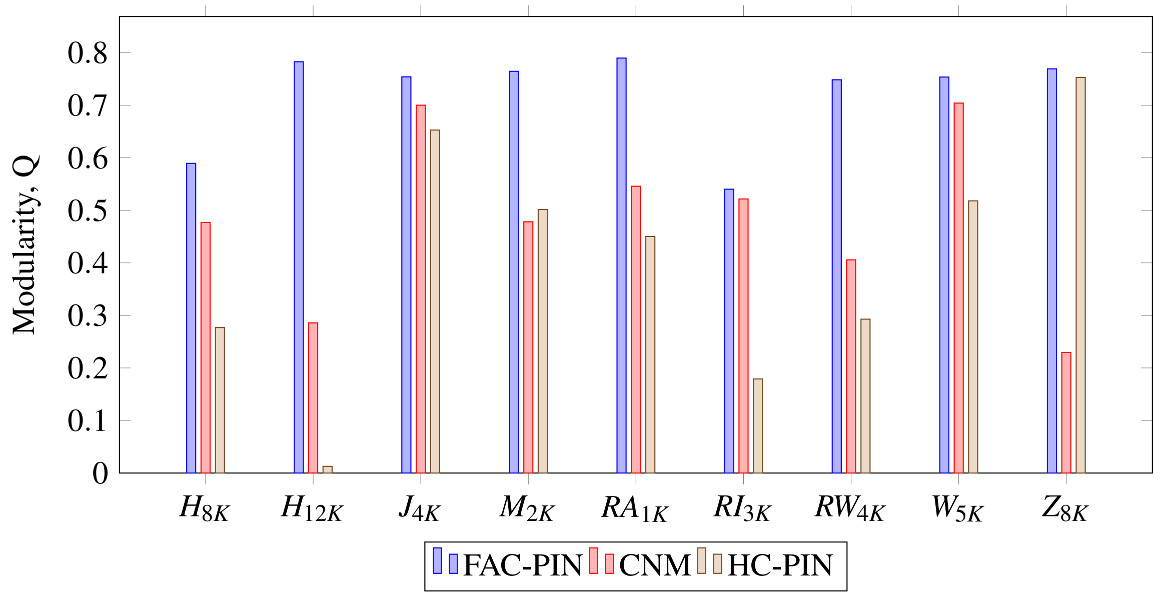

Figure 4.2: Modularity comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish

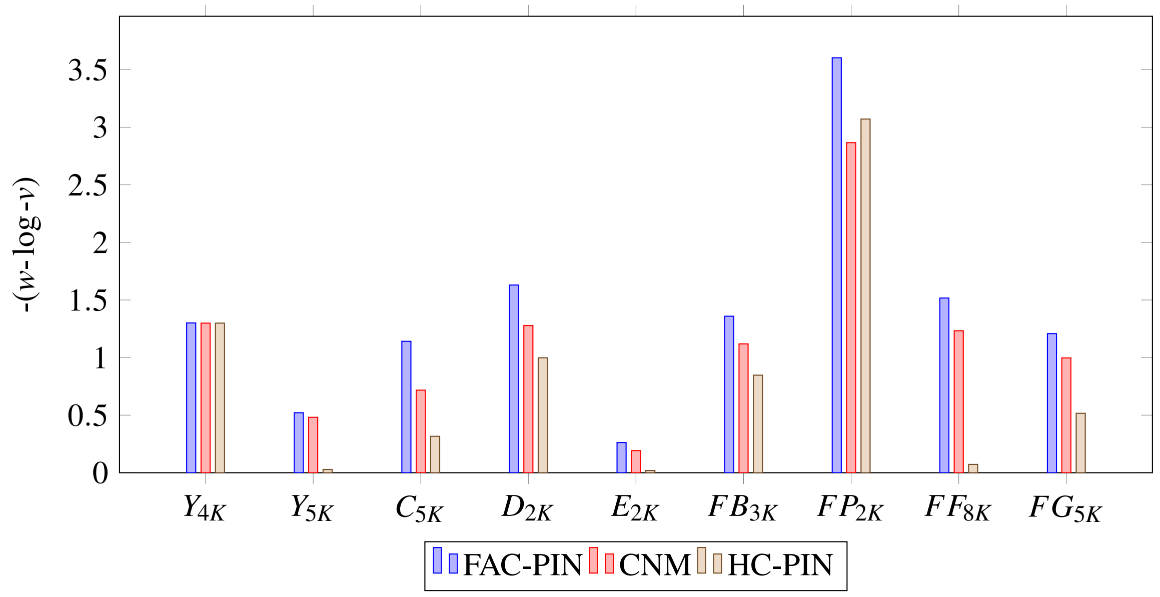

Figure 4.3: -(*w*-log-*v*) comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog

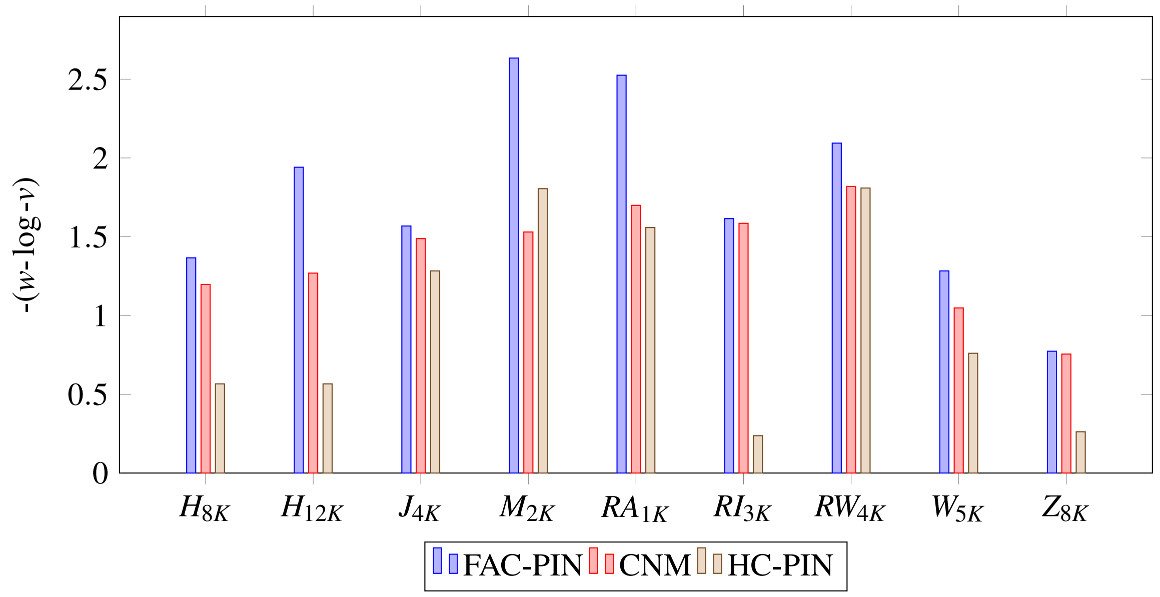

Figure 4.4: -(*w*-log-*v*) comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish

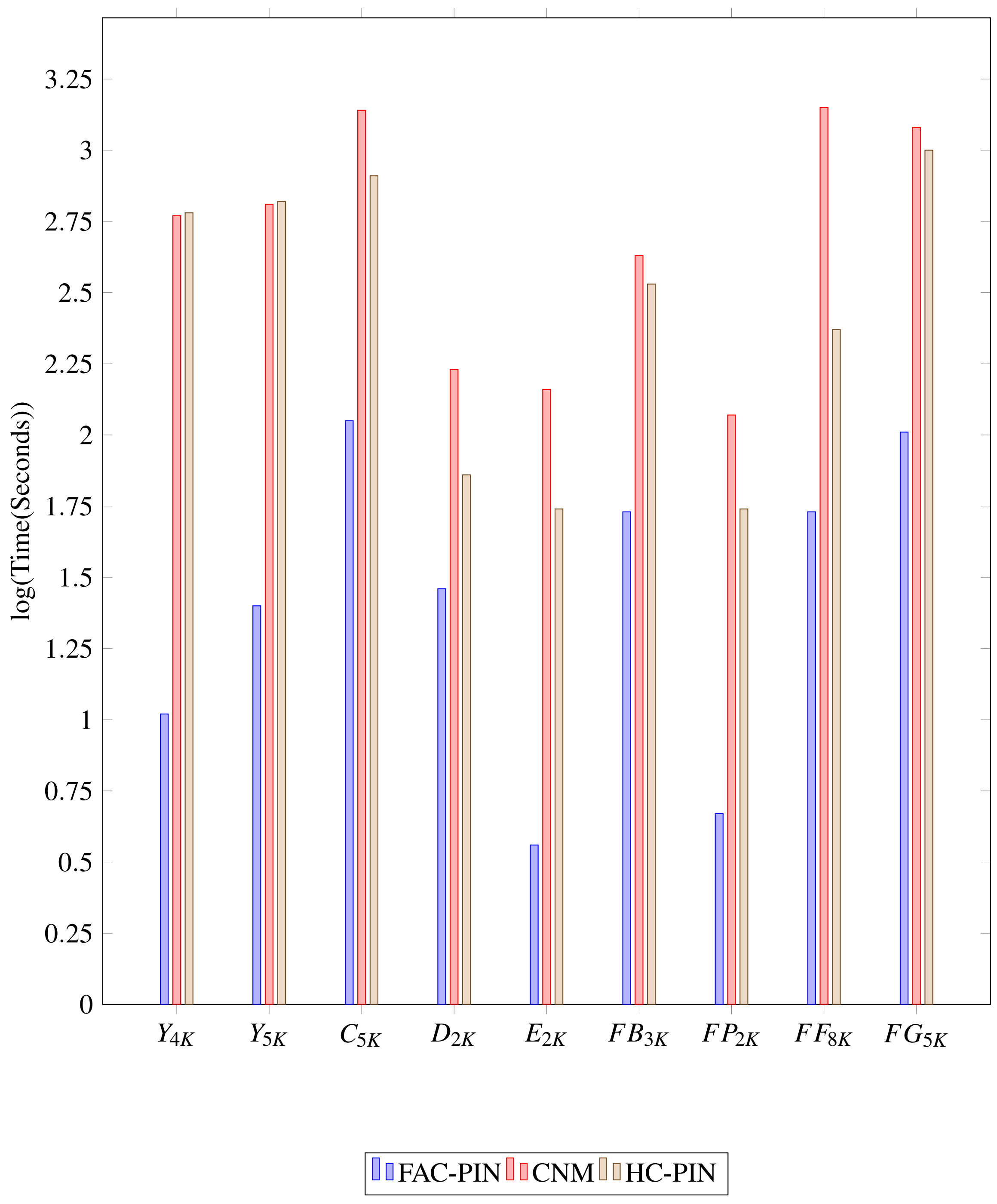

Figure 4.5: Log-log plot of execution time comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog

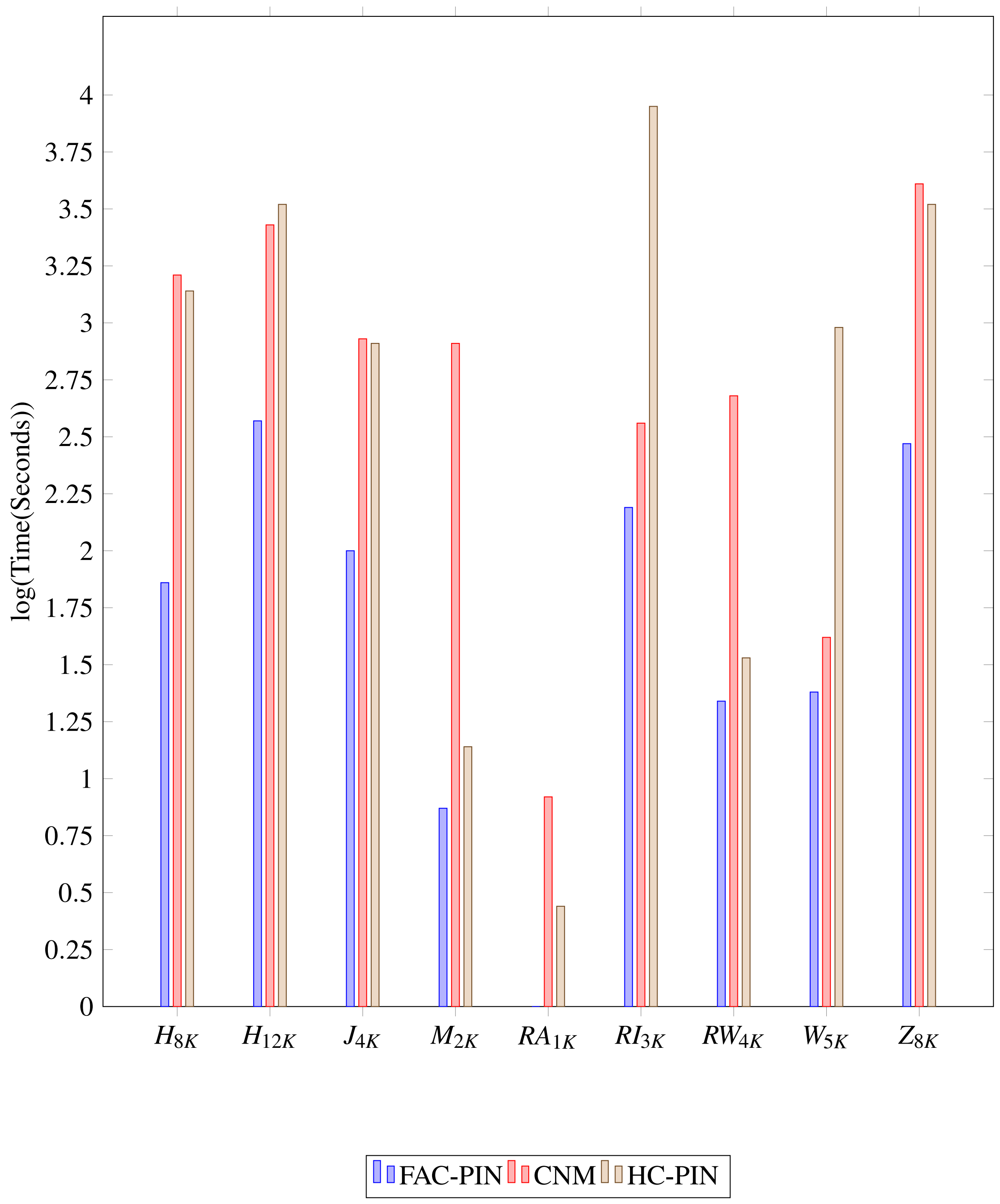

Figure 4.6: Log-log plot of execution time comparison among *FAC-PIN*, *CNM* and *HC-PIN* algorithms for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish

Table 4.6: Time comparison results of *FAC-PIN*, *CNM* and *HC-PIN* for Baker's yeast, Cattle, Dog, E.coli, Finch bird, Flowering plant, Fruit fly and Frog (in seconds).

Algorithms	Y_{4K}	Y_{5K}	C_{5K}	D_{2K}	E_{2K}	FB_{3K}	FP_{2K}	FF_{8K}	FG_{5K}
FAC-PIN	10.44	25.12	112.7	29.02	3.66	54.04	4.77	54.85	104.05
CNM	598.96	645.03	1394.71	170.38	144.94	426.86	119.40	1428.98	1205.93
HC-PIN	607.25	663.50	818.91	73.42	55.02	340.67	55.02	234.69	1004.13

Table 4.7: Time comparison results of *FAC-PIN*, *CNM* and *HC-PIN* for Human, Jungle fowl, Mouse, Rat, Rice, Round worm, Wild boar and Zebra fish (in seconds).

Algorithms	H_{8K}	H_{12K}	J_{4K}	M_{2K}	RA_{1K}	RI_{3K}	RW_{4K}	W_{5K}	Z_{8K}
FAC-PIN	73.63	372.59	102.02	7.44	1.00	156.15	22.25	24.54	295.74
CNM	1653.28	2743.28	856.06	155.33	8.46	370.57	484.25	41.68	4102.23
HC-PIN	1381.23	3372.31	830.45	13.99	2.78	9001.37	34.52	953.92	7461.35

terms of either the number of proteins or the number of interactions, the difference between the performances of FAC-PIN and HC-PIN (or CNM) also increase greatly. In Tables 4.2 and 4.4, it is found that for small baker's yeast protein interaction network (Y_{4K}) modularity Q and $w\text{-}\log\text{-}v$ of three algorithms are almost same. The reason behind that the most of the vertices of this PIN are distributed in almost clustered manners (please see Figures 4.7, 4.8, 4.9 and 4.10). This is also true in Tables 4.6, 4.7 and Figures 4.5, 4.6 showing the execution times (seconds) and log-log plot of the time comparisons of the three algorithms. Clearly FAC-PIN is much faster than the other two methods (even large gaps between log-log plots of FAC-PIN algorithms and HC-PIN, CNM algorithms), and again, the difference in performance increases as either the number of proteins or the number of interactions increases. All experiments were performed on an Intel machine (Core TM i7-2600, 3.400 GHz, CPU with 8 GB RAM).

4.3 Protein Complex Discovery

We validated our results by comparing the communities detected by FAC-PIN with a list of protein complexes obtained from the MIPS (***M****unich* ***I****nformation center for* ***P****rotein*

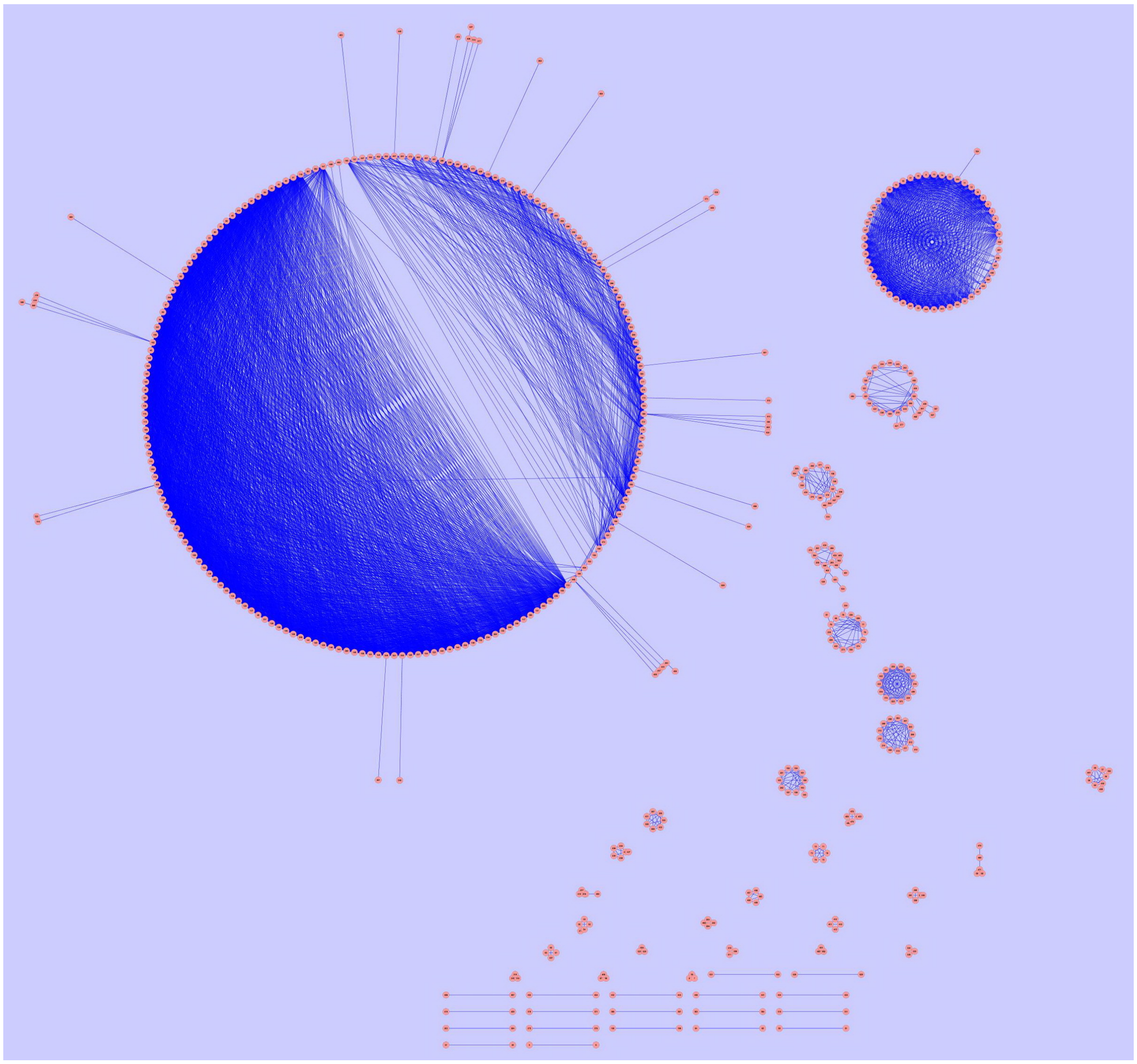

Figure 4.7: Protein Interaction Network of Baker's yeast (Y_{4K})

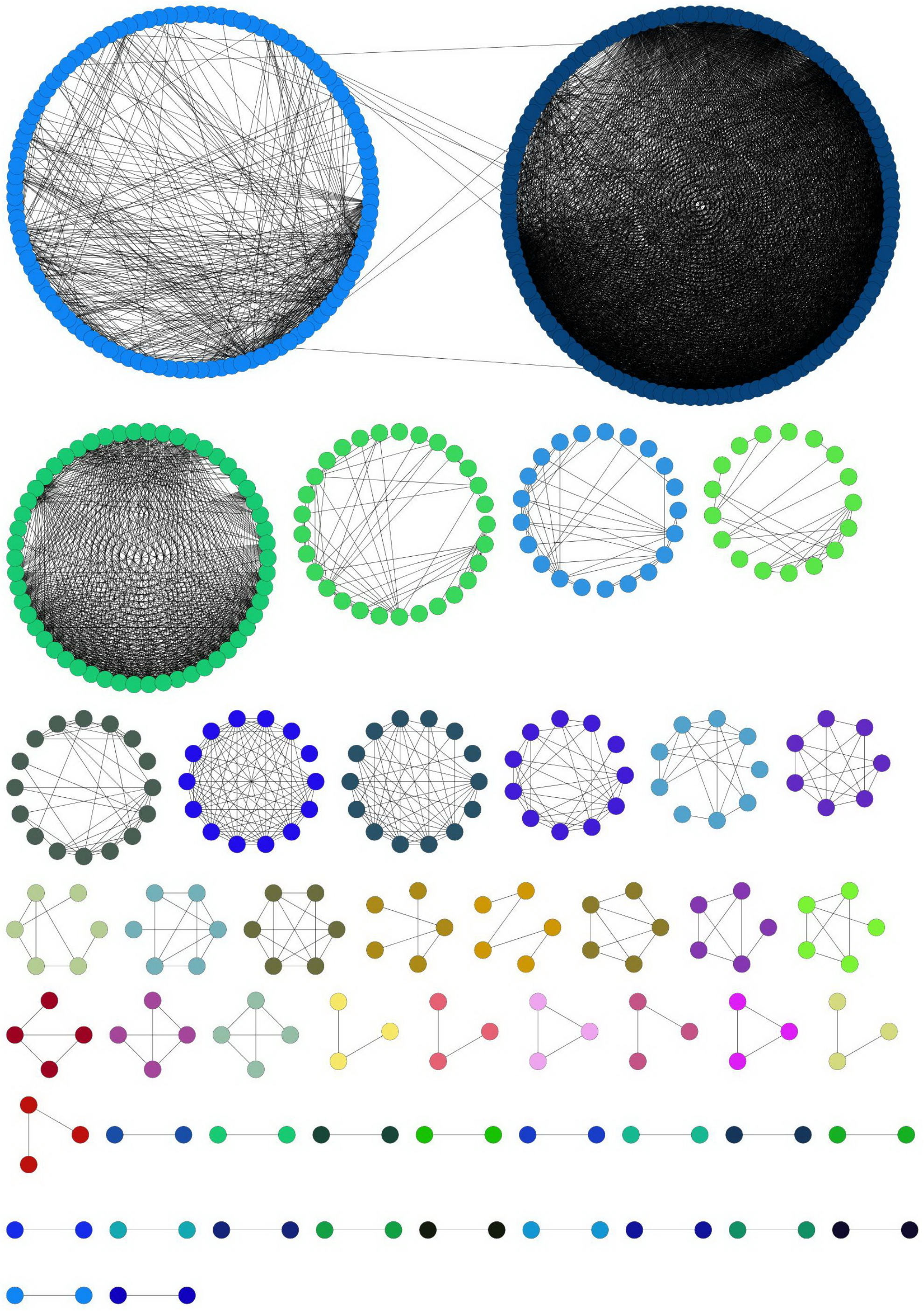

Figure 4.8: Clustered Network of Baker's yeast (Y_{4K}) using FAC-PIN algorithm

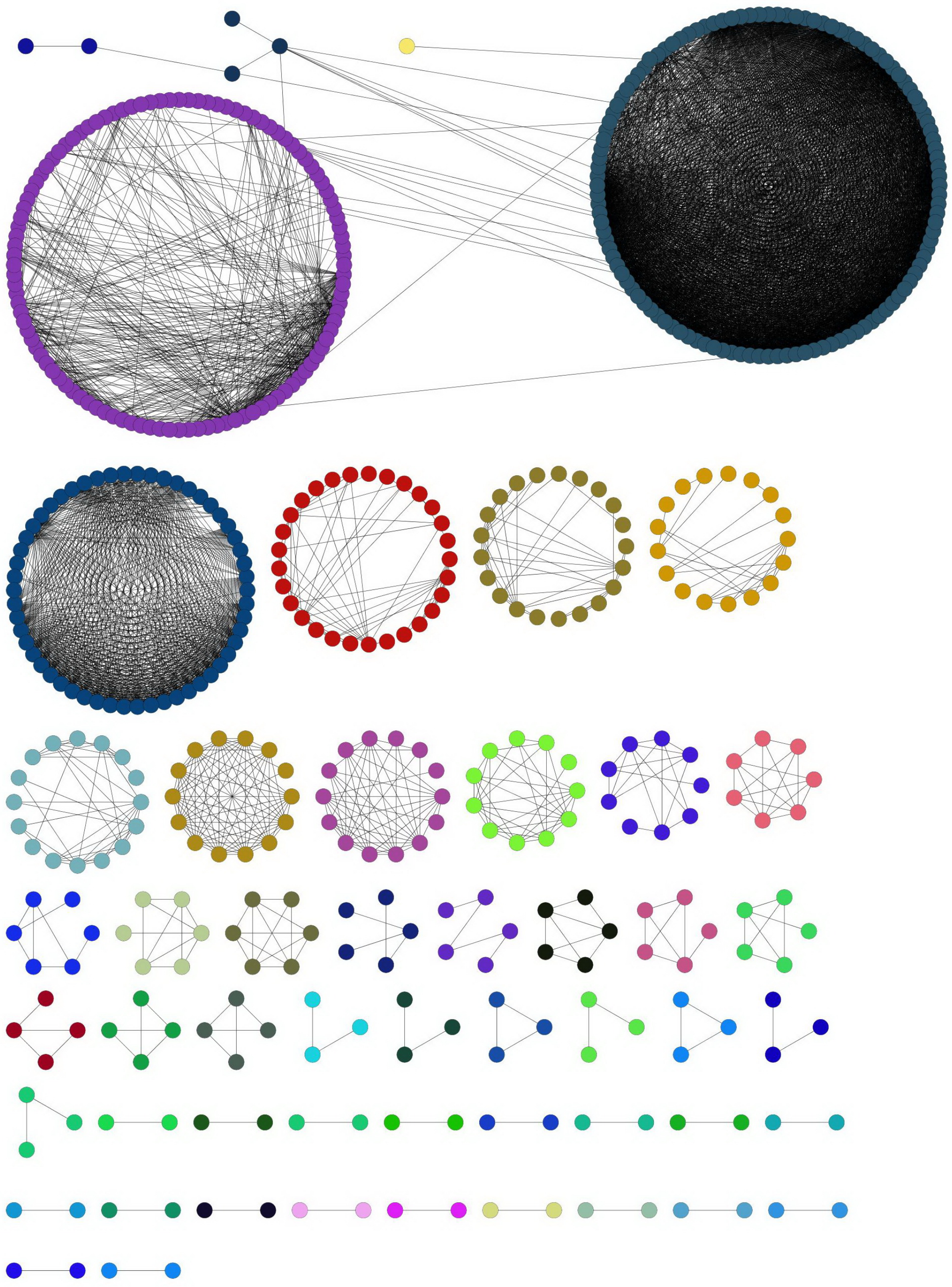

Figure 4.9: Clustered Network of Baker's yeast (Y_{4K}) using CNM algorithm

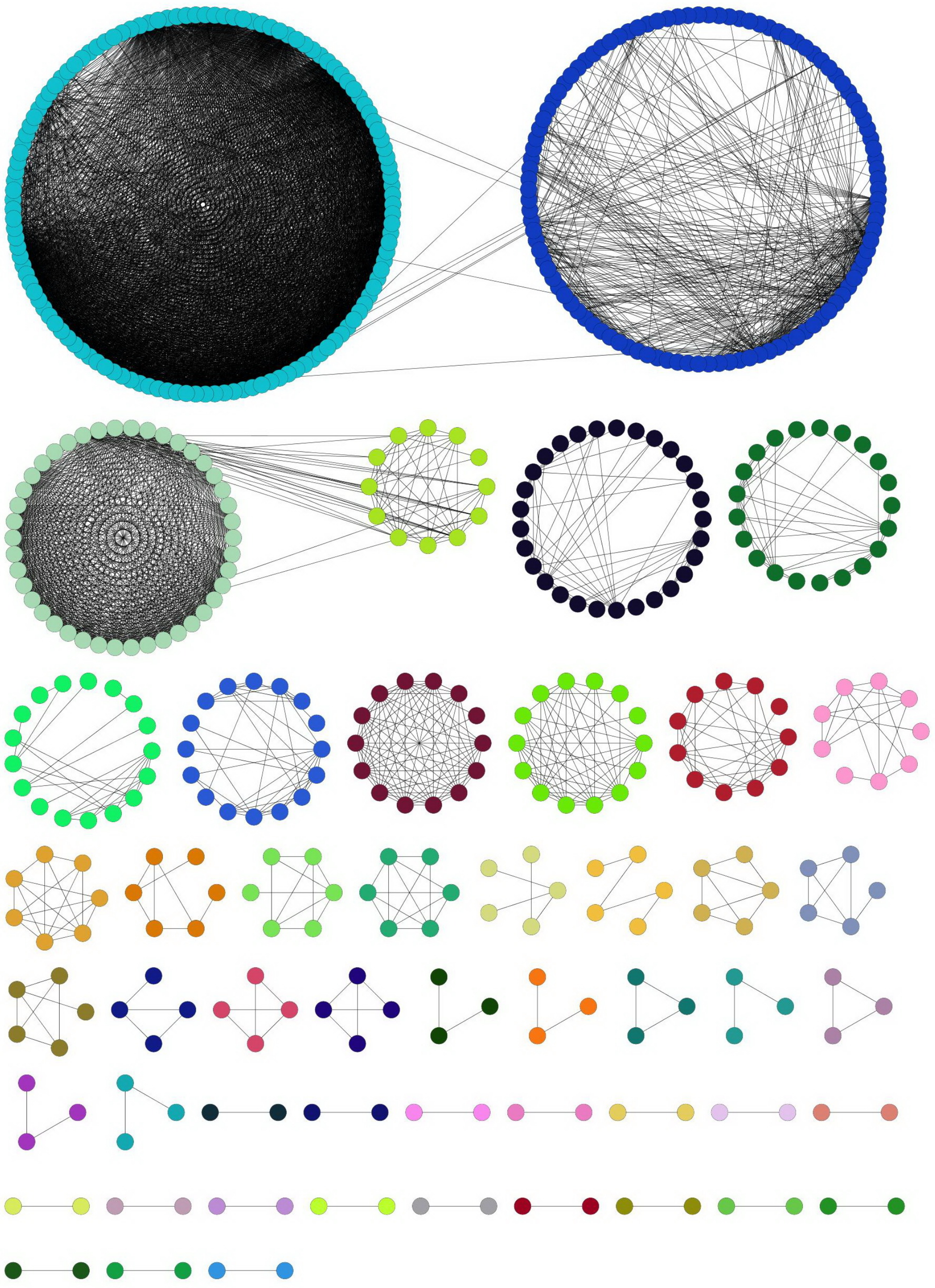

Figure 4.10: Clustered Network of Baker's yeast (Y_{4K}) using HC-PIN algorithm

***S**equences*) databases, which we consider as a *gold standard* data. Our validation were done only for four species which we could download corresponding complexes from MIPS. For *Baker's yeast*'s PIN, we obtained complexes from the MIPS *Comprehensive Yeast Genome Database-CYGD*[4]. For the PINs of *Human, Mouse* and *Rat*, the corresponding complexes were downloaded from the MIPS *Comprehensive Resource of Mammalian Protein Complexes- CORUM* [5].

We proceeded similarly to Laarhoven et al. [18] for baker's yeast and considered only the known complexes (i.e., not those obtained by computational means) containing at least three proteins. Since FAC-PIN generates non-overlapping communities, we considered only complexes which are at the bottom of the MIPS hierarchy of complexes and subcomplexes. The unconfirm complexes, that is those in category 550, were excluded.

Before computing complex validation testing, we generated the clusters from FAC-PIN, HC-PIN and CNM algorithms for baker's yeast, human, mouse and rat using same proteins which were present in their corresponding protein complexes. We used modularity function Q as scoring function to select the best clustered network for each species.

The validation proceeds by determining the degree of overlap between the communities identified by FAC-PIN and the protein complexes; in effect, determining how effectively a community matches a known complex. We used the same scoring scheme used in [2,6,18,30]. The overlapping score, $O(C,K)$, between a community C and a known complex K is given as:

$$O(C,K) = \frac{|C \cap K|^2}{|C| \times |K|} \tag{4.3}$$

[4]`ftp://ftpmips.gsf.de/yeast/catalogues/complexcat/`

[5]`http://mips.helmholtz-muenchen.de/genre/proj/corum/`

Community C and known complex K are considered a match whenever $O(C,K) \geq \tau$, where $0 < \tau \leq 1$ is the matching threshold. We have a perfect match only if $O(C,K) = 1$. Threshold value $\tau = 0.2$ is used in [2,6,30], whereas $\tau = 0.25$ is in [18]. They considered the threshold values where the average number of matched predicted clusters and protein complexes are higher. We used both values of τ in our complex validation. After determining the overlapping scores between all communities and all known complexes for a given PIN, we determined the ability of FAC-PIN to correctly classify the known complexes. The reason for doing this is that a given complex K_1 may match many communities with different degrees of overlap, while another complex K_2 may match with a single community only. Hence, we computed the *Specificity*, the *Sensitivity*, and the *F-score*, as our measures of accuracy, and which are given below

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.4}$$

$$Specificity = \frac{TP}{TP + FP} \tag{4.5}$$

$$F\text{-score} = \frac{2 \times specificity \times sensitivity}{specificity + sensitivity} \tag{4.6}$$

where, *TP* (true positive) is the number of the identified communities C matched by the known complexes K, *FN* (false negative) is the number of known complexes that are not matched by the communities, and *FP* (false positive) is the total number of the identified communities C minus *TP*. Tables 4.8, 4.9, 4.10 and 4.11 show the comparative results of the *Specificity SP*, the *Sensitivity SN*, and the F-score FS of FAC-PIN, HC-PIN and CNM for baker's yeast, human, mouse and rat respectively. In these Tables, we have also showed the number of proteins n that were used in complex validation process, the number of complexes C, the average size of complex $\mid \bar{C} \mid$, the number of clusters k, the average size of cluster $\mid \bar{k} \mid$, the number of matched k_m and perfectly matched clusters k_{m_p} for easy comparison.

Table 4.8: Comparison of the *Specificity*, *Sensitivity* and *F-score FAC-PIN*, *CNM* and *HC-PIN* for *Baker's yeast*

					Computed results						
n	C	$\mid \bar{C} \mid$	Algorithms	τ	k	$\mid \bar{k} \mid$	k_m	k_{m_p}	SN	SP	FS
1237	267	4.63	*FAC-PIN*	0.2	285	4.34	165	12	0.61	0.58	0.594
			CNM		300	4.12	99	5	0.37	0.33	0.348
			HC-PIN		111	11.14	56	3	0.21	0.51	0.297
			FAC-PIN	0.25	285	4.34	148	12	0.55	0.52	0.534
			CNM		300	4.12	99	5	0.37	0.33	0.348
			HC-PIN		111	11.14	55	3	0.21	0.50	0.296

Table 4.9: Comparison of the *Specificity*, *Sensitivity* and *F-score FAC-PIN*, *CNM* and *HC-PIN* for *Human*

					Computed results						
n	C	$\mid \bar{C} \mid$	Algorithms	τ	k	$\mid \bar{k} \mid$	k_m	k_{m_p}	SN	SP	FS
2555	575	4.44	*FAC-PIN*	0.2	607	4.21	291	8	0.50	0.48	0.489
			CNM		639	3.99	255	5	0.44	0.40	0.419
			HC-PIN		119	21.47	58	3	0.10	0.49	0.166
			FAC-PIN	0.25	607	4.21	450	8	0.78	0.74	0.759
			CNM		639	3.99	198	5	0.34	0.31	0.324
			HC-PIN		119	21.47	52	3	0.09	0.44	0.149

Table 4.10: Comparison of the *Specificity*, *Sensitivity* and *F-score FAC-PIN*, *CNM* and *HC-PIN* for *Mouse*

					Computed results						
n	C	$\mid \bar{C} \mid$	Algorithms	τ	k	$\mid \bar{k} \mid$	k_m	k_{m_p}	SN	SP	FS
935	460	2.03	*FAC-PIN*	0.2	568	1.64	335	13	0.72	0.59	0.64
			CNM		605	1.54	338	6	0.73	0.56	0.63
			HC-PIN		151	6.19	75	3	0.163	0.50	0.25
			FAC-PIN	0.25	568	1.64	312	13	0.68	0.55	0.61
			CNM		605	1.54	332	6	0.71	0.55	0.61
			HC-PIN		151	6.19	75	3	0.163	0.50	0.25

In Tables 4.8, 4.9, 4.10 and 4.11 we see that FAC-PIN identifies communities whose average sizes (column 7) are closer to the average sizes of known proteins complexes (column 3) whereas HC-PIN and CNM yield larger averages of cluster sizes. The consequence

Table 4.11: Comparison of the *Specificity*, *Sensitivity* and *F-score FAC-PIN*, *CNM* and *HC-PIN* for *Rat*

					Computed results						
n	C	$\lvert \bar{C} \rvert$	Algorithms	τ	k	$\lvert \bar{k} \rvert$	k_m	k_{m_p}	SN	SP	FS
557	328	1.69	*FAC-PIN*	0.2	389	1.42	167	7	0.51	0.43	0.466
			CNM		475	1.17	138	3	0.42	0.29	0.343
			HC-PIN		117	4.76	38	1	0.116	0.33	0.172
			FAC-PIN	0.25	389	1.42	128	7	0.39	0.33	0.356
			CNM		475	1.17	109	2	0.33	0.23	0.271
			HC-PIN		117	4.76	25	1	0.076	0.22	0.113

of this is that smaller FAC-PIN communities produce higher accuracy (*Specificity*, *Sensitivity* or *F-score*) in the great majority of cases. This because, most of the known complexes are small, and that the smaller a complex the higher the accuracy. In particular, we obtain a larger number of perfectly matched complexes to communities with FAC-PIN than with HC-PIN or CNM.

4.4 Functional Module Identification

We validated our results by comparing the communities detected by FAC-PIN with a list of functional modules obtained from the *Gene Ontology Annotation* databases, which we consider as a *gold standard* data. Our validation were done only for two species (*Baker's yeast* and *rat*) which we could download corresponding functional modules from Gene Ontology Annotation databases. For *Baker's yeast*'s PIN, we obtained modules from the *Yeast Genome* database[6]. For the PIN of *Rat*, the corresponding modules were downloaded from the *Genome Ontology Annotation* database[7].

Like *overlapping score* of protein complex validation process, we used a measure called *p-value* to detect significant modules. the *p-value* is calculated from cumulative distribu-

[6]http://downloads.yeastgenome.org/curation/literature/go_slim_mapping.tab
[7]http://www.geneontology.org/GO.downloads.annotations.shtml

tion function, *cdf* of *hypergeometric distribution*:

$$p\text{-}value = 1 - \sum_{j=0}^{k-1} \frac{\binom{|F_i|}{j}\binom{|V|-|F_i|}{|M|-j}}{\binom{|V|}{|M|}} \tag{4.7}$$

Where F_i is a functional category mapped to module M. The proteins in functional category F_i are considered as true predictions (physical functional modules), the proteins in module M are considered as positive predictions (computer identified functional modules) and the common proteins of F_i and M are considered as true positive predictions k. It is understood as the probability that at least k proteins in M are included in F_i. Low *p-value* indicates that the positive predicted module closely corresponds to the true predicted module, because the network has a lower probability to produce community by chance [16].

For this reason we have used a term called *cutoff* value to detect the significant modules. If the p-value of a positive predicted module (computer generated modules) (in short it is called ppm) is less than cutoff value, we considered it as significant module. This process was also used by Wang et al. [30] for validating their HC-PIN algorithm. Both Wang et al. [30] and we considered the cutoff value is 0.05. Besides *p-value*, We used also another two important aspects- *Recall* and *Precision* to estimate the performance of the algorithms for detecting functional modules [30]. Recall is the fraction of the true positive predictions out of all true prediction and *Precision* is the fraction of true positive predictions out of all positive predictions. The formulas of Recall and Precision follow as below-

$$Recall = \frac{|M \cap F_i|}{|F_i|} \tag{4.8}$$

$$Precision = \frac{|M \cap F_i|}{|M|} \tag{4.9}$$

Recall indicates how effectively proteins with the same functional category in the network are extracted and *Precision* illustrated how consistently proteins in the same module are annotated [30]. In general, the module with large size will have higher *Recall* and module with smaller size will have higher *Precision*. Thus, a harmonic mean of *Recall* and *Precision*, *f-measure* is defined as follow-

$$f\text{-}measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{4.10}$$

The accuracy of an algorithm is defined as the average of *f-measure* of the significant modules generated by it. In the experiment, we considered only positive predicted modules which have three or more proteins. On the other hand, we have separated functional modules into three ontologies- *Biological Process* abbreviated as *BP*, *Molecular Function* abbreviated as *MF* and *Cellular Component* abbreviated as *CC*. For each ontology, we computed functional module test. As well as we computed the functional module test on the combination of all three ontologies. The results of functional modules validation for baker's yeast is shown in Table 4.12. In the Table-4.12, we have listed the number of identified modules whose p-values falls within $p\text{-}value < E-15$, $[E-15 \ to \ E-10]$, $[E-10 \ to \ E-5]$, $[E-5 \ to \ E-1]$, average -log(p-value) as $-\overline{\log p}$ and average $f-measure$ as $\overline{fm}$. As well as we have listed the number of ppm as k_p, the number of ppm with three or more proteins as k_{3p}, the average size of ppm as $\mid \overline{k_p} \mid$ and the number of significant clusters as k_s.

From the Table-4.12, we found that FAC-PIN gave more accurate results than HC-PIN and CNM algorithm for baker's yeast because of higher average *f-measure*. As well as the percentage value of the column-6 of Table 4.12 shows that, FAC-PIN found more functional modules than other two algorithms.

Table 4.12: Functional enrichment of the identified Modules which comprises of three or more proteins of *Baker's yeast*

Algorithm	k_p	k_{3p}	$\mid \overline{k_p} \mid$	Ontology	k_s	k_s for different p-value				$-\overline{\log p}$	$\overline{fm}$
						$< E - 15$	[E-15 to E-10]	[E-10 to E-5]	[E-5 to E-1]		
FAC-PIN	702	522	7.18	Combine	241 (46.2%)	64	17	10	150	4.66	0.165
				BP	168	10	8	9	141	4.14	0.171
				CC	168	60	10	8	90	4.52	0.133
				MF	168	8	14	9	137	3.94	0.106
HC-PIN	218	159	23.13	Combine	72 (45.3%)	4	4	10	54	3.17	0.024
				BP	61	2	4	3	52	2.52	0.025
				CC	61	3	2	9	47	3.09	0.012
				MF	61	3	4	3	51	3.21	0.019
CNM	609	226	8.27	Combine	19 (8.4%)	5	0	2	12	4.45	0.031
				BP	12	0	0	2	10	3.93	0.035
				CC	12	3	0	1	8	4.33	0.040
				MF	12	0	0	0	12	3.90	0.032

In Tables-4.13 , we have listed the number of identified larger modules ($size \geq 20$), percentage of significant larger modules, average -log(p-value) as $-\overline{\log p}$ and average *f-measures* as $\overline{fm}$. FAC-PIN, HC-PIN and CNM algorithms successfully identifies all larger modules for all types ontologies. On the other hand, in the Table-4.14, we have listed the number of identified smaller modules ($size \leq 6$), percentage of significant smaller modules, average -log(P-value) and average *f-measures*. In this case, our FAC-PIN algorithm outperformed HC-PIN and CNM algorithms.

Table 4.13: Performance comparison of clustering algorithms for the identification of large-sized ($size \geq 20$) modules of *Baker's yeast*

Ontologies	Algorithms	$k_{\geq 20}$	$k_{\geq 20s}$	%	$-\overline{\log p}$	$\overline{fm}$
BP	*FAC-PIN*	24	24	100%	6.20	0.1702
	HC-PIN	2	2	100%	5.52	0.0055
	CNM	4	4	100%	4.61	0.0249
CC	*FAC-PIN*	24	24	100%	7.13	0.1337
	HC-PIN	2	2	100%	6.09	0.0033
	CNM	4	4	100%	5.95	0.0368
MF	*FAC-PIN*	24	24	100%	6.20	0.1059
	HC-PIN	2	2	100%	5.19	0.0148
	CNM	4	4	100%	5.72	0.0258

We have showed the functional module testing results in the Tables 4.15, 4.16 and 4.17

Table 4.14: Performance comparison of clustering algorithms for the identification of small-sized (*size* ≤ 6) modules of *Baker's yeast*

Ontologies	Algorithms	$k_{\leq 6}$	$k_{\leq 6s}$	%	$\overline{-\log p}$	$\overline{fm}$
BP	*FAC-PIN*	101	101	100.00%	4.63	0.0702
	HC-PIN	13	11	84.61%	4.22	0.0158
	CNM	5	4	80.00%	4.54	0.0121
CC	*FAC-PIN*	101	101	100.00%	5.07	0.0133
	HC-PIN	13	12	92.31%	5.04	0.0129
	CNM	5	5	100.00%	4.95	0.0118
MF	*FAC-PIN*	101	101	100.00%	4.28	0.0148
	HC-PIN	13	13	100.00%	4.19	0.0108
	CNM	5	4	80.00%	4.22	0.0115

for Rat. Accept cellular component (CC) ontology, FAC-PIN algorithm generated the better positive predicted modules because of higher average *f-measure* compare to other two algorithms. For cellular component, CNM algorithm found the large modules having more than 20 proteins more accurately than FAC-PIN and HC-PIN algorithms. It affected the total results of cellular component. Similar to baker's yeast, FAC-PIN, HC-PIN and CNM algorithms perform excellent to identify the large modules. On the other hand, for small modules, FAC-PIN identifies the modules more accurately then others. But for molecular function ontology, though the accuracy is higher, unfortunately all small modules of FAC-PIN algorithms are not significant. Only 95% small modules are matched or significant. Despite the cellular component ontology result, FAC-PIN outperformed the HC-PIN and CNM algorithms.

4.5 Conclusion

The modularity Q and $w\text{-}\log\text{-}v$ results of sixteen different species for FAC-PIN, HC-PIN and CNM algorithm indicates that FAC-PIN algorithm finds the more dense subgraph. The reason behind it: FAC-PIN algorithm solves the problem of clustering the vertices of degree one. As well as our proposed FAC-PIN algorithm ran faster than other algorithm. For all

Table 4.15: Functional enrichment of the identified Modules which comprises of three or more proteins of *Rat*

Algorithm	k_p	k_{3p}	$\mid \overline{k_p} \mid$	Ontology	k_s	k_s for different p-value				$\overline{-\log p}$	$\overline{fm}$
						$< E-15$	[E-15 ~ E-10]	[E-10 ~ E-5]	[E-5 ~ E-1]		
FAC-PIN	627	268	2.09	Combine	93 (34.7%)	10	7	4	72	4.66	0.165
				BP	93	10	7	4	72	5.47	0.004
				CC	72	2	3	5	62	5.11	0.012
				MF	86	2	14	4	66	5.21	0.018
HC-PIN	218	269	23.13	Combine	93 (34.5%)	4	4	5	80	3.17	0.024
				BP	70	0	0	2	68	5.37	0.002
				CC	90	3	3	5	79	4.92	0.011
				MF	82	4	0	5	73	5.06	0.008
CNM	576	261	2.27	Combine	78 (28.8%)	5	3	6	64	4.45	0.031
				BP	65	0	0	1	64	5.32	0.004
				CC	66	4	1	3	58	4.68	0.014
				MF	65	0	3	5	57	5.22	0.012

Table 4.16: Performance comparison of clustering algorithms for the identification of large-sized ($size \geq 20$) modules of *Rat*

Ontologies	Algorithms	$k_{\geq 20}$	$k_{\geq 20s}$	%	$\overline{-\log p}$	$\overline{fm}$
BP	*FAC-PIN*	8	8	100.00%	6.38	0.0034
	HC-PIN	14	14	100.00%	6.17	0.0007
	CNM	15	15	100.00%	5.93	0.0032
CC	*FAC-PIN*	8	8	100.00%	6.96	0.0084
	HC-PIN	14	12	85.71%	6.02	0.0053
	CNM	15	15	100.00%	5.48	0.0094
MF	*FAC-PIN*	8	8	100.00%	7.67	0.0171
	HC-PIN	14	10	71.42%	6.65	0.0039
	CNM	15	12	80.00%	6.67	0.0101

Table 4.17: Performance comparison of clustering algorithms for the identification of small-sized ($size \leq 6$) modules of *Rat*

Ontologies	Algorithms	$k_{<20}$	$k_{<20s}$	%	$\overline{-\log p}$	$\overline{fm}$
BP	*FAC-PIN*	45	45	100.00%	6.85	0.0002
	HC-PIN	45	45	100.00%	6.67	0.0001
	CNM	34	34	100.00%	6.83	0.0001
CC	*FAC-PIN*	45	45	100.00%	5.80	0.0029
	HC-PIN	67	67	100.00%	5.51	0.0027
	CNM	34	32	94.12%	5.79	0.0025
MF	*FAC-PIN*	45	43	95.55%	5.44	0.0010
	HC-PIN	66	66	100.00%	5.70	0.0010
	CNM	34	28	82.35%	5.84	0.0007

species it took several minutes even several seconds to find the best clustered PINs. On the other hand, HC-PIN and CNM algorithms took more time (even two or three hours).

In complex validation testing, FAC-PIN identifies the protein complexes more accurately than other algorithms (CNM and HC-PIN). it is known that sensitivity determines the fraction of complexes which are matched with computer generated communities. The sensitivity of FAC-PIN algorithm is higher than the others. It means FAC-PIN algorithm can identify the more complexes than CNM and HC-PIN. On the other hand, specificity determines the fraction of communities which are matched with experimentally defined complexes. In this case, FAC-PIN also outperformed the CNM and HC-PIN algorithms. *F-score* determines the overall performance of the test. Both specificity and sensitivity of FAC-PIN algorithm is higher than others, so its overall performance is also better than others.

Except, cellular component ontology of rat species, FAC-PIN can identify the functional modules for all ontologies of baker's yeast and rat efficiently.

We cannot compute the protein complex validation and functional module identification testing for all species due to the limitation of time and shortage of computer memory. Cattle, Frog, Human, Rice, Wild boar and Zebra fish species have large PINs which cause the memory limitation for performing protein complex prediction and functional module identification process.

Though, limitation of time and computer physical memory, our FAC-PIN algorithm outperforms existing fast algorithm for PIN: HC-PIN and popular community detection algorithm: CNM in respect to execution time, dense community structures, complex validation and function module prediction tests.

Chapter 5

Conclusion and future work

This thesis presents an efficient algorithm (FAC-PIN) for finding the communities more accurately than others. FAC-PIN algorithm is designed on the basis of newly defined premetric *Relative vertex-to-vertex clustering value*. It computes likelihood value of a vertex u to merge with another vertex v to form a cluster. This premetric deals with the problem of clustering the vertices of degree one efficiently.

Our designed FAC-PIN algorithm solves the problem of classifying the vertices of degree one efficiently by using relative vertex-to-vertex clustering value. After solving this problem, we have found that testing (scoring functions) results represent that our proposed FAC-PIN algorithm gives better dense subgraph than other algorithms for sixteen different species.

Our proposed FAC-PIN algorithm runs faster than known best and fast hierarchical algorithms: HC-PIN and CNM. The running time complexity of HC-PIN and CNM algorithms are $O(m\bar{d}^2)$ and $O(mh\log_2 n)$ respectively. In PINs, the number of interactions are quite larger than the number of proteins. So, for large PINs, these two algorithm require more execution time. Whereas, our proposed FAC-PIN algorithm needs $O(n\bar{d}^2)$ times to find the clusters or communities from PINs. The number of proteins are very less than the number of interactions, it helps FAC-PIN algorithm to run faster than others.

From the computational results of evaluating clusters or communities, complex validation, functional modules identification testing, we can say that, our proposed FAC-PIN algorithm is more efficient than CNM and HC-PIN algorithms. As well as it is faster algorithm than other twos due to fastest execution time over HC-PIN and CNM algorithm.

Still several new doors are opened to continue research on FAC-PIN and relative vertex-to-vertex clustering values. We have listed them as follows:

- Some protein interaction networks are weighted. There is a door opened to improve FAC-PIN algorithm and relative vertex-to-vertex clustering value for weighted protein interaction networks.

- We have designed our algorithm only for protein networks. It can be redesigned for social networks.

- FAC-PIN algorithm has been designed on the basis of *Relative vertex-to-vertex clustering value*. The FAC-PIN algorithm can be improved by introducing ΔQ [7], $Q_{SigBound}$ [1] etc.

- FAC-PIN algorithm has been designed by using the concept of first order neighbouring concept. It can be improvised by introducing the concept of second order neighbouring concept [].

- In FAC-PIN algorithm, we considered a random values for threshold α for clustering any PIN. There is another door opened for determining best α for PIN of each species.

- FAC-PIN algorithm has been designed based on the concept of *Preferential attach-*

ment property. *Active protein* [31] concept can be introduced in FAC-PIN algorithm which will produce improved clustered PINs.

Bibliography

[1] M. Altaf-Ul-Amin, Y. Shinbo, K. Mihara, K. Kurokawa, and S. Kanaya. Development and implementation of an algorithm for detection of protein modulees in large interaction networks. *BMC Bioinformatics*, 7(207):207 – 219, 2006.

[2] G.D. Bader and C.W. Hogue. An automated method for finding molecular modulees in large protein interaction networks. *BMC Bioinformatics*, 7:1 – 27, 2003.

[3] C. Branden and J. Tooze. *Introduction to Protein Structure*, volume 2. Garland Publishing: New York, 1993.

[4] A. Bruce, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the cell*. Garland Science, 4th edition, 2002.

[5] J. Chen and B. Yuan. Detecting functional modules in the yeast proteinprotein interaction network. *BMC Bioinformatics*, 22(18):1 – 12, 2006.

[6] H.N. Chua, K. Ning, W.-K. Sung, Leong H.W., and L. Wong. Using indirect protein-protein interaction for protein module prediction. *ournal of Bioinformatics and Computational Biology*, 6(6):1 – 13, 2008.

[7] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Nature*, 453(7191):1 – 6, 2005.

[8] S.V. Dongen. Graph clustering by flow simulation. *Ph.D. Thesis, University of Utrecht, Natherland*, 2000.

[9] S. Fortunato. Community detection in graphs. *Elsevier Physics Reports*, 486:86 – 197, 2010.

[10] A.-C. Gingras, R. Aebersold, and B. Raught. Advances in protein complex analysis using mass spectrometry. *Journal of Physiol*, 563.1:11–21, 2005.

[11] L Giot, J.S. Bader, C. Brouwer, A. Chaudhuri, B. Kuang, Y. Li, Y.L. Hao, C.E. Ooi, B. Godwin, and E. Vitols. A protein interaction map of *Drosophila melanogaster*. *Science*, 302:1727 – 1736, 2003.

[12] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. *Proceedings of Natural Academy of Science, USA*, 99:7821 – 7826, 2002.

[13] A. Gursoy, O. Keskin, and Nussinov R. Topological properties of protein interaction networks from a structural perspective. *Biochemical Society Transactions*, 36:1398 – 1403, 2008.

[14] J. Hallinan. Gene duplication and hierarchical modularity in intracellular interaction networks. *Biosystems*, 74(3):10 – 15, 2004.

[15] Hopfield J.J. Leibler S. Murray A.W. Hartwell, L.H. From molecular to modular cell biology. *Nature*, 402:47 – 52, 1999.

[16] E.C. Kenley and Y.-R. Cho. Detecting protein complexes and functional modules from protein interaction networks: A graph entropy approach. *Proteomics*, 11:1116 – 1121, 2011.

[17] A.D. King, N. Pržulj, and I. Jurisica. Protein complex prediction via cost-based clustering. *BMC Bioinformatics*, 20(17):3013 – 3020, 2004.

[18] T.V. Laarhoven and E. Marchiori. Robust community detection methods with resolution parameter for module detection in protein protein interaction networks. *Springer-Verlag Berlin*, 7632:1 – 13, 2012.

[19] M. Li, J.X. Wang, and J.E. Chen. A fast agglomerative algorithm for mining functional modules in protein interaction networks. *Proceedings of First International Conference in BioMedical Engineering and Informatics (BMEI)*, pages 3 – 7, 2007.

[20] X.L. Li, S. Tan, C. Foo, and S. Ng. Interaction graph mining for protein modulees using local clique merging. *Genome Informatics*, 16:260 – 269, 2006.

[21] F. Luo, Y. Yang, C.-F. Chen, R. Chang, J. Zhou, and R.H. Scheuermann. Modular organization of protein interaction networks. *BMC Bioinformatics*, 23(3):207 – 214, 2007.

[22] M.E.J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review*, 69(066133):1 – 5, 2003.

[23] G. Palla, I. Derény, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814 – 818, 2005.

[24] L. Pauling, R.B. Corey, and H.R. Branson. The structure of proteins; two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of Natural Academy of Science, USA*, 37(4):20 – 26, 1951.

[25] P. Pei and A. Zhang. A 'seed-refine' algorithm for detecting protein modulees from protein interaction data. *IEEE Transcation of Nanobioscience*, 6(1):43 – 50, 2007.

[26] F. Radicchi, C. Castellano, and F. Cecconi. Defining and identifying communities in networks. *Proceedings of Natural Academy Of Sciences USA*, 101(9):2658 – 2663, 2004.

[27] E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, and A.-L. Barabśsi. Hierarchical organization of modularity in metabolic networks. *Science*, 297:1551 – 1555, 2002.

[28] J.S. Richardson. The anatomy and taxonomy of protein structure. *Advances in Protein Chemistry*, 34, 1981.

[29] V. Spirin and L.A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of Natural Academy of Science USA*, 100(21):12123 – 12128, 2003.

[30] J. Wang, Chen Li, M., and Y. J., Pan. A fast hierarchical clustering algorithm for functional modules discovery in protein interaction networks. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 8(3):607 – 620, 2011.

[31] J. Wang, X. Peng, Q. Xiao, M. Li, and Y. Pan. An effective method for refining predicted protein complexes based on protein activity and the mechanism of protein complex formation. *BMC Systems Biology*, 7(28):30 – 53, 2013.

[32] H. Xiong, P. Tan, and V. Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. *Proceeding of the Third IEEE International Conference on Data Mining (ICDM)*, pages 221– 232, 2003.

[33] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Proceedings of IEEE 12th International Conference on Data Mining*, 12:745 – 754, 2012.

[34] S. Yook, Z. Olvai, and A.L. Barabsi. Functional and topological characterization of protein interaction networks. *Protenomics*, 4:928 – 942, 2004.

Vita Auctoris

Mohammad Rahman was born in 1984 in Dhaka, Bangladesh. He received his Bachelors degree from the University of Dhaka in Computer Science and Engineering in 2005 and masters degree from Bangladesh University of Engineering and Technology (BUET) in Information and Communication Technology in 2010. He has 2 journal papers and 4 conference papers. His research interests include pattern recognition and bioinformatics.