MODULE IDENTIFICATION FOR BIOLOGICAL NETWORKS

A Dissertation

by

YIJIE WANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2015

Major Subject: Electrical Engineering

ABSTRACT

Advances in high-throughput techniques have enabled researchers to produce large-scale data on molecular interactions. Systematic analysis of these large-scale interactome datasets based on their graph representations has the potential to yield a better understanding of the functional organization of the corresponding biological systems. One way to chart out the underlying cellular functional organization is to identify functional modules in these biological networks. However, there are several challenges of module identification for biological networks. First, different from social and computer networks, molecules work together with different interaction patterns; groups of molecules working together may have different sizes. Second, the degrees of nodes in biological networks obey the power-law distribution, which indicates that there exist many nodes with very low degrees and few nodes with high degrees. Third, molecular interaction data contain a large number of false positives and false negatives.

In this dissertation, we propose computational algorithms to overcome those challenges. To identify functional modules based on interaction patterns, we develop efficient algorithms based on the concept of block modeling. We propose a subgradient Frank-Wolfe algorithm with path generation method to identify functional modules and recognize the functional organization of biological networks. Additionally, inspired by random walk on networks, we propose a novel two-hop random walk strategy to detect fine-size functional modules based on interaction patterns. To overcome the degree heterogeneity problem, we propose an algorithm to identify functional modules with the topological structure that is well separated from the rest of the network as well as densely connected. In order to minimize the impact of the existence of noisy interactions in biological networks, we propose methods to detect conserved functional modules for multiple biological networks by integrating the topological and orthology information across different biological networks. For every algorithm we developed, we compare each of them with the state-of-the-art algorithms on several biological networks. The comparison results on the known gold standard biological function annotations show that our methods can enhance the accuracy of predicting protein complexes and protein functions.

ACKNOWLEDGEMENTS

Many thanks go to my advisor Dr. Xiaoning Qian for his excellent guidance, overwhelming passion for scientific problems and his continuous encouragement during my PhD study. I would also like to thank Dr. Byung-Jun Yoon, Dr. Jean-Francois Chamberland-Tremblay, and Dr. Sing-Hoi Sze for guiding my research for the past several years and helping me to develop my background in bioinformatics.

I would like to thank Shaogang Ren, Meng Lu, Amin Ahmadi Adl, Siamak Zamani, Meltem Apaydin and Chung-Chi Tsai for their research collaboration and discussion of our research work.

I would also like to thank my parents. They were always supporting me and encouraging me with their best wishes.

Last but the most importantly, I would like to thank my wife Xiaoqing Huang, whose patience, love and understanding make all this possible.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION*

What is the next big wave in technology? Different people may have different opinions. Let us look at the Internet giant — Google's next move. Along with the lines of self-driving cars and smart glasses, Google's newest venture is called California Life Company, whose goal is to extend human's life by 20 to 100 years. It seems unreal, however, the request to live a little bit longer has been in demand from the beginning of the human society. Actually, the investigations of one kind of flatworm have shed light on the possibilities of alleviating aging in human cells since researchers have demonstrated that the flatworm can overcome the aging process and could potentially live forever [103]. However, to enforce the mission impossible, several fundamental questions need to be answered in advance, such as what is the molecular mechanism of an organism and how the molecular mechanism controls the activities within the organism.

To unveil the mystery of those basic biological problems through understanding their underlying cellular mechanism, it is indispensable to look into the Deoxyribonucleic acid (DNA), which is a molecule that stores the genetic instructions used in all biological processes of all known living organisms. Human Genome Project (HGP) has achieved tremendous success in determining the DNA sequence and recognizing and mapping genes of the human genome based on both their physical and functional responsibilities. However, HGP collaborated all research pioneers around the world and still costed 13 years and $3 billions, which illustrates how difficult to sequence a general genome in the last decades. With the help of fast development

*Fig 1.1 in this chapter is reprinted with permission from "Wisdom of crowds for robust gene network inference" by Daniel Marbach, James C Costello, Robert Kffner, Nicole M Vega, Robert J Prill et al, Nature Method, 9(8): 796-804, 2012, Copyright 2012 by Nature Method.

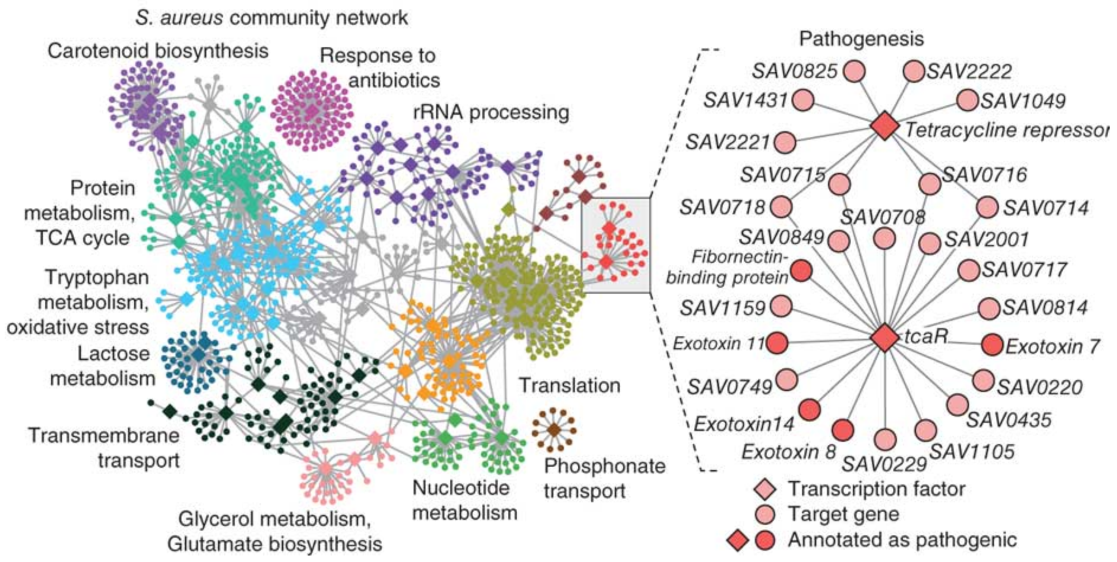


Figure 1.1: An example for module identification for a gene co-expression network.

of high-throughput technologies, nowadays, obtaining biological data, such as genomics data, proteomics data and molecular interactions becomes more efficient and less expensive. Some widely used high-throughput technologies are next-generation sequencing, mass spectrometry, yeast two-hybrid assays and microarrays. Due to the availability and diversity of the high-throughput technologies, there are tons of biological data generated every day, which brings biologists into a brand-new era with the biological data they have never had in the past.

The dramatically increasing generation of biological data enables us to better understand biological systems. Identifying functional modules in biological systems is a fundamental way to comprehend the functional organizations of the corresponding biological systems and interpret their underlying mechanisms. Biologically, a functional module in a biological system consists of a group of biological units, which perform similar functions. In this dissertation, we focus on identifying functional modules in biological systems, which are modeled as biological networks constructed from interactome data. Fig. 1.1 shows an example of module identification in a gene co-expression network [59]. The modules in the networks are identified barely based on the topological structures. The functional organization and topological relationships between modules are clearly illustrated through module identification.

1.1 Biological networks

Complex biological systems can be represented and investigated as networks. In general, a node of a biological network represents a protein or a gene and an edge indicates the association between nodes. Based on different purposes and different biological systems, different types of networks are generated. For example, there are protein-protein interaction networks, transcript-transcript association networks (gene co-expression networks) and DNA-protein interaction networks (Gene regulatory networks).

For all kinds of theses biological networks, a principle, called guilt-by-association, is widely used. Guilt-by-association declares that the nodes (biological units) in the biological networks, which are connected by an edge, are more likely to perform the same function than nodes (biological units) not linked together. Therefore, it is possible to predict the function of an unknown node (biological unit) through the functions of its topological neighborhood, which have been validated either by chemical experiments or biological experts.

Because complicated biological systems are abstracted into biological networks, basic biological problems are converted to network related problems. Therefore, the challenges over the next decades are to make use of the information existing in the biological networks to answer fundamental biological problems [78], such as functional organization and robustness of biological systems. Here we briefly introduce two kinds of widely used biological networks, which are protein interaction networks and gene co-expression networks.

1.1.1 Protein interaction networks

Proteins are large molecules composed of one or more connected amino acid residues. They carry out diverse functions within living organisms. Biologically, proteins rarely act alone. Through protein-protein interactions, groups of proteins are organized together to facilitate diverse fundamental molecular processes within a cell.

Protein-protein interactions in a cell construct a protein interaction network, where nodes represent proteins and edges represent physical protein-protein interactions. Thanks to the high-throughput technologies, all physical protein-protein interactions in a cell can be screened in one test. There are many ways to detect the physical binding interactions. The most widely used high-throughput methods of detecting the physical protein-protein interactions are yeast two-hybrid screening (Y2H) [38] and affinity purification coupled to mass spectrometry (AP-MS) [114]. Y2H was proposed by Fields and Song in 1989 [88]. Pairwise protein interactions with binary weights can be inferred by Y2H. In Y2H, the transcription factor is separated into two fragments, which are called binding domain and activating domain. The binding domain is fused onto a protein of interesting (referred as the bait protein) and the activating domain is fused onto another protein (referred as the prey protein). If the bait protein and prey protein interacts with each other, then the activating domain is brought to the transcription start site, which incurs the occurrence of the transcription of reporter genes. If those two proteins do not interact, then there is no occurrence of the transcription of reporter genes. Based on the occurrence of the transcription of reporter genes, the interactions between proteins can be identified. The limitation of Y2H is that the number of identified protein-protein interactions is low due to the loss of transient protein interactions in purification steps. AP-MS consists two steps. In the AP (affinity purification) step, a protein of interest, called bait, is affinity caught in a matrix. The bait protein is passed through the matrix, the protein, interacting with the bait protein, is retained due to interaction with the bait. After purification, proteins can be analyzed by MS (mass spectrometry), which is a chemistry technique that helps to determine the amount and type of chemicals in a sample [114]. There are other profiling techniques to detect protein-protein interactions, however, the interactions identified by those methods are noisy [73].

There are many public protein interaction databases available for researchers and scientists. Some databases such as BioGrid [101], DIP [90] and IntAct [41] contain protein interaction networks of many different species. Some databases only maintain protein interaction networks of specific species, such as HPRD [82] (a database for human protein interaction network) and FlyBase [27] (a database for fruit fly protein interaction network).

1.1.2 Gene co-expression networks

Similar to protein interaction networks, gene co-expression networks are also network with symmetric interactions, where each node represents a gene and each edge indicates the similarity of the co-expression patterns of a pair of genes with respect to samples. Gene co-expression networks are of biological interest because co-expressed genes may be controlled by the same transcriptional regulatory mechanism, or the same pathway or protein complex.

The construction of a gene co-expression networks follows a two-step approach. In the first step, the similarity between every pair of gene expression data is calculated. Then in the second step, edges with small similarities are filtered out by setting a threshold value. The input data for formulating a gene co-expression network is stored in a matrix format. For example, in a microarray experiment, we can obtain the gene expression values of m genes and n samples, then the input data is a $m \times n$ matrix, which is called the expression matrix. Each row in the expression matrix implies the gene expression pattern of that gene. In the first step, we can estimate the similarity of the gene expression patterns between pairwise genes through computing the similarity between two row vectors. Pearson's correlation coefficient, mutual information, spearman's rank correlation coefficient and Euclidean distance are the four mostly used co-expression measures [102]. After the calculation of the similarity scores, we obtain an $m \times m$ similarity matrix, each element of which shows how similar two genes change together with respect to expression levels. In the second step, the elements of the $m \times m$ similarity matrix are dichotomized based on a certain threshold. The dichotomized matrix is the adjacency matrix of the gene co-expression network. "1" in the adjacency matrix denotes two genes are correlated under the same samples or conditions, and "0" otherwise.

1.2 Challenges

Although there are many algorithms developed to identify functional modules in different types of biological networks, efficient algorithms still need to be devised to detect modules with better accuracy and less computational time. Here we summarize the challenges we encountered.

1.2.1 What is the good definition of a functional module?

Intuitively, based on interactome data, if two nodes interact with each other, they are more likely to share the same cellular functionalities than nodes that do not interact. Thus, densely connected subnetworks in a given network can be viewed as potential functional modules. Based on this idea, many algorithms have been successfully applied to identify functional modules in biological networks by detecting "higher than expected connectivity" subnetworks based on modularity optimization [75, 74] or random walk on graphs [105, 91, 93, 68].

However, in addition to densely connected modules in biological networks, such as protein complexes, there are other topological structures that may possess important cellular functionalities. Again, based on interactome data, the nodes that interact with similar sets of other nodes in a given network also intuitively have a higher probability of sharing the similar functionalities compared to the nodes that do not share any interacting partners or neighbors [80, 86, 64]. These nodes may not directly interact with each other but they still work towards similar cellular functionalities and hence should belong to the same modules. It is well known that transmembrane proteins, such as receptors in signal transduction cascades, tend to interact with cytoplasmic proteins as well as with extra-cellular ligands, but rarely interact with themselves [80]. To identify such types of functional modules, many state-of-the-art block modeling module identification algorithms have been proposed recently [67, 66, 86, 35]. But those algorithms suffer from the prohibitive computational complexity due to the inherent combinatorial complexity of the block modeling problem. Therefore, more efforts need to be made to develop algorithms that can efficiently identify functional modules based on nodes interaction patterns.

1.2.2 Degree heterogeneity

The degrees of nodes in a biological network obey the power-law distribution: therefore, there exist many nodes with low degrees and few nodes with high degrees, which is called degree heterogeneity. Due to degree heterogeneity, it is hard to design module identification algorithms with the presence of the nodes of very low degrees. Several algorithms are developed for power-law networks but most of them have not been applied on biological networks [20], and most algorithms designed for biological networks do not consider the degree heterogeneity problem [80, 86, 64, 75, 74, 91, 93].

1.2.3 Biological experiment noise

Currently, the biological interactions in the biological networks generated by high-throughput experiments are very noisy. For example, it is well known that there are lots of false positive and false negative interactions presented in the protein interaction networks [21]. Therefore, module identification simply based on individual biological networks may not be able to yield robust and accurate results. We may need to appropriately integrate other available information in addition to network topology, such as sequence and function similarity, to repress the noise. Comparative analysis of multiple biological networks enable us to borrow topological information across networks and incorporate the corresponding homological information to reduce the influence of noise. Many algorithms [22, 94, 46, 39, 116] have been developed to find conserved modules in multiple networks.

1.3 Our contributions

We develop efficient computational algorithms to overcome the challenges discussed in the previous section.

1.3.1 Module identification based on interaction pattern

To discover modules with richer topological structures, we devise a novel subgradient method with heuristic path generation [109] to accelerate the optimization process for the block modeling formulation [80]. To overcome the resolution problem [30] of the block modeling formulation [80], we propose an algorithm SLCP2 [110] based on two-hop random walk strategy to identify fine-size functional modules considering the interaction patterns. Additionally, we propose an non-negative matrix factorization (NMF) based framework for more general module identification problems by sparse regularized. We discuss these algorithms in more details in Chapter 3.

1.3.2 Overcoming the degree heterogeneity

We devise a new local algorithm that can identify densely connected modules with the existence of low-degree nodes. This new algorithm consists two local steps guided by optimization principles. The algorithm first searches for a low-conductance set near a node, which is well separated from the rest of the network and then identifies the densest sub-network in the low-conductance set to get rid of the low-degree nodes. This new algorithm with such a principle-guided seed expansion and shrinking procedure outperforms state-of-the-art module identification algorithms in biological networks. We present this new algorithm and experimental results in Chapter 5.

1.3.3 Identifying conserved modules in multiple networks

We incorporate homological information across networks to repress the noise in each individual networks. We extend the block modeling formulation and two-hop random walk strategy to search functional modules based on interaction patterns in pairwise biological networks [112], which are discussed in details in Chapter 4. Furthermore, in Chapter 5, we develop an novel algorithm to identify conserved modules in multiple networks, which are topological cohesive and possess many-to-many homological correspondences.

Before we getting into the details of our developed module identification algorithms, we first introduce the background, basic mathematics notations, and related work to the module identification problem.

2. RELATED WORK*

In this chapter, we review the state-of-the-art module identification algorithms for both individual networks and multiple networks. For individual networks, we first go through methods of identifying topological cohesive modules and then review algorithms that search functional modules based on interaction patterns. For multiple networks, we survey the state-of-the-art methods for pairwise and multiple networks, respectively, based on how they representing multiple networks [89, 54, 99, 46, 22, 94, 39, 116].

2.1 Module identification for topological cohesive modules

To identify topological cohesive modules, many algorithms have been successfully applied based on modularity optimization [75, 74]. Additionally, several algorithms based on Markov random walk on networks also have been proposed recently. For example, Markov CLustering (MCL) algorithm is one of such module identification algorithms for biological network analysis by iteratively implementing "Expand" and "Inflation" operations on the transition matrix of the underlying Markov chain of random walk [26]. Regularized MCL (RMCL) [91, 93] further extends the original MCL algorithm to penalize the large cluster size at each iteration to obtain more balanced modules with a similar number of nodes within them. Other formulations based on Markov random walk, including finding low conductance sets [105], also can be applied in module identification, which is in fact similar to normalized cut problems [115] in graph partitioning to minimize the normalized cut size across modules. Recently, several overlapping module identification methods have been developed to

*Fig 2.3 in this chapter is reprinted with permission from "A novel subgradient-based optimization algorithm for blockmodel functional module identification" by Yijie Wang and Xiaoning Qian, BMC Bioinformatics, 14(Suppl 2): S23, 2013, Copyright 2013 by BMC Bioinformatics.

detect densely connected modules that may overlap with each other in networks. For example, ClusterOne (CONE) [68] can be viewed as the overlapping version of normalized cut. LinkComm (Link community) [1] formulates the overlapping module identification in an innovative framework to implement the hierarchical clustering on edge graph representations, which reveals hierarchical and overlapping organization of networks.

In this section, we review the representative definitions of a topological cohesive module, such as modularity and conductance, and the corresponding algorithms. Assuming we have a biological network in graph representation $G(V, E)$, where V is the set of $|V| = n$ nodes representing proteins or genes, and $E = \{e_{ij}\}$ is the set of $|E| = m$ edges, which suggest the physical interactions or correlations. The network G can be presented by an adjacency matrix A, whose element A_{ij} of which equals 1 if there is an edge between nodes i and j and 0 otherwise. We only consider unweighted networks with binary edge weights. D is a diagonal matrix with $D_{ii} = \deg(i)$, where $\deg(i) = \sum_j A_{ij}$ is the degree of node i. The goal of module identification is to detect a group of nodes, which perform similar functions or possess identical properties, barely based on the network topologies.

2.1.1 Community detection based on modularity

Community detection is one of the major directions of functional modules identification. Community detection aims to identify groups of nodes that are densely connected inside and loosely connected outside. Intuitively, a good community in network $G(V, E)$ should be a group of nodes C such that there are many more edges between the nodes in C than from the nodes in C to nodes in $V - C$. One important definition that has been intensively studied for community detection is called modularity [74].

2.1.1.1 The definition of modularity

Basically, modularity expresses the relationship between the actual connectivity inside a group of nodes C and the expected connectivity in C. The formal mathematical formulation of the modularity of C is

$$Q_C = \sum_{ij} \left[A_{ij} - P_{ij}\right] \delta\left(g_i, g_j\right), \tag{2.1}$$

where g_i denotes the community that node i belongs to and $\delta(s,t) = 1$ if $s = t$ and 0 otherwise. P_{ij} is the expected number of edges between nodes i and j. Matrix P can be considered as the weighted adjacency matrix of a null model, which has the same number of nodes in G. P should satisfy the following constraints. First, P is symmetric, which implies $P_{ij} = P_{ji}$. Second, $Q_C = 0$ when all nodes are placed in a single group. Setting all $g_i \in C$, we have

$$\sum_{ij} \left[A_{ij} - P_{ij}\right] = 0 \Rightarrow \sum_{ij} A_{ij} = \sum_{ij} P_{ij} = 2m. \tag{2.2}$$

Physically, the equation means that the expected number of edges in the entire null model equals the number of actual edges in G. Additionally, we require the degree distribution of the null model is approximately the same to the original network G. Hence, we need

$$\sum_{j} P_{ij} = d_i, \tag{2.3}$$

where d_i is the degree of node i. One widely used null model, which satisfies the conditions above is

$$P_{ij} = \frac{d_i d_j}{2m}. \tag{2.4}$$

Based on the definition of modularity, we can identify k non-overlapping communities $\{G_1, G_2, ..., G_k\}$ in G by maximizing the corresponding modularity. For this k-way partition, node i can be assigned to $g_i = \{1, 2, ..., k\}$. Then the maximization of modularity can be written as

$$\max : \sum_{ij} \sum_{g_i, g_j} [A_{ij} - P_{ij}] \, \delta(g_i, g_j) \,. \tag{2.5}$$

Obviously, (2.5) is a combinatorial optimization problem, which is computational intractable by exhaustively searching all possible k partitions in G. However, many algorithms [74, 13, 2] have been proven effective. Prof. Mark Newman, who originally proposed the definition of modularity, developed a method to approximate the solution of (2.5) [74] by using the eigen-system of matrix $A - P$. Blondel devised a greedy method to handle large-scale networks with good solution quality [13]. [2] extended the column generation methods for mixture integer programming to find the exact solution of (2.5).

2.1.1.2 The algorithm based on eigenvectors

Here, we briefly review the community detection method using eigenvectors by Newman [74]. Following [74], we define a binary $n \times k$ community assignment matrix X, where the ith row indicates the membership of node i and the jth column presents the jth community. Formally,

$$X_{ij} = \begin{cases} 1 & \text{if node } i \text{ belongs to community } j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

Note that every row sum of X is 1 and the columns of X are orthogonal.

Noticing that the δ function in (2.5) is equivalent to

$$\delta\left(g_i, g_j\right) = \sum_{l=1}^{k} X_{il} X_{jl}. \tag{2.7}$$

Then (2.5) can be written as

$$\begin{aligned} Q =& \sum_{ij} \sum_{g_i, g_j} \left[A_{ij} - P_{ij}\right] \delta\left(g_i, g_j\right) \\ =& \sum_{i,j=1}^{n} \sum_{l=1}^{k} \left[A_{ij} - P_{ij}\right] X_{il} X_{jl} \\ =& \mathrm{Tr}(X^T B X), \end{aligned} \tag{2.8}$$

where $B = A - P$. There is an implicit constraint for the above problem, which is X is a binary assignment matrix with $\sum_j X_{ij} = 1$ and $\sum_l X_{li} X_{lj} = 0, \forall i \neq j$.

Because B is a symmetric matrix, hence, it can be diagonalized $B = U \Lambda U^T$, where $U = (u_1|u_2|...)$ is the matrix of eigenvectors of B and Λ is a diagonal matrix of eigenvalues $\Lambda_{ii} = \beta_i$. B may not be positive semi-definite matrix, which means β_i may be negative. To alleviate the influence of the negative eigenvalues, we do the following transformation

$$\begin{aligned} Q &= \mathrm{Tr}(X^T B X) \\ &= \mathrm{Tr}(X^T U \Lambda U^T X) \\ &= \alpha n + \mathrm{Tr}(X^T U (\Lambda - \alpha I) U^T X). \end{aligned} \tag{2.9}$$

Here we make use of the fact that $\mathrm{Tr}(X^T X) = n$ and α is related to the negative eigenvalues. We further remove $n - p$ eigenvectors, whose corresponding eigenvalues smaller than α, then the above equation becomes

$$\begin{aligned} Q &= \alpha n + \mathrm{Tr}(X^T U(\Lambda - \alpha I)U^T X) \\ &\approx \alpha n + \mathrm{Tr}(X^T U_p(\Lambda_p - \alpha I_p)U_p^T X) \\ &= \alpha n + \sum_{l=1}^{k}\sum_{j=1}^{p}\left(\sum_{i}^{n}\sqrt{\beta_j - \alpha}U_{ij}X_{il}\right)^2 \end{aligned} \tag{2.10}$$

We define n p-dimensional vectors $r_i = \sqrt{\beta_j - \alpha}U_{ij}$ to characterize each node i in G. Then $\sum_{j=1}^{p}\left(\sum_{i}^{n}\sqrt{\beta_j - \alpha}U_{ij}X_{il}\right)^2 = \sum_{j=1}^{p}\left(\sum_{i\in G_l}[r_i]_j\right)^2 = |R_l|^2$, where R_l is the sum of all p-dimensional vectors that belongs to community l. Finally, the modularity Q can be approximated by

$$Q \approx \alpha n + \sum_{l=1}^{k}|R_l|^2. \tag{2.11}$$

Therefore the maximization of modularity is converted to clustering the nodes in G into groups so as to maximize the magnitudes of the vectors R_l, which is called vector partition problems. The k-means algorithm can be easily applied to solve the vector partition problem.

2.1.1.3 The resolution problem

Identification of communities for biological and social networks using modularity has been proved effective by many researchers [16, 55]. However, [30] pointed out that using modularity can not resolve some small-size meaningful modules. Therefore, identification of modules like protein complexes in protein interaction networks becomes the bottle-neck for modularity based algorithms.

2.1.2 Community detection based on conductance

Another definition that can help us identify community structures in G is conductance [6]. Conductance measures how fast a random walk on G converges to the stationary distribution. For a set of nodes C, its conductance is defined as

$$\phi(C) = \frac{|E(C, \bar{C})|}{\min\{\text{vol}(C), \text{vol}(\bar{C})\}}, \tag{2.12}$$

where $\text{vol}(C) = \sum_{i \in C} \deg(i)$ and $|E(C, \bar{C})|$ denotes the connections between sets C and $\bar{C} = V - C$. Finding S with the minimal conductance is called conductance minimization. In this section, we will discuss several algorithms [95, 6, 26], which are closely related to conductance.

2.1.2.1 Normalized cut

For a set C, the normalized cut [95] is defined as

$$N_{cut}(C) = \frac{|E(C, \bar{C})|}{\text{vol}(C)}. \tag{2.13}$$

Comparing (2.13) with (2.12), obviously, only the denominator is different. When dividing G into large enough k parts $\{V_1, V_2, ..., V_k\}$, reasonably assuming at each partition $\text{vol}(V_i) < \text{vol}(\bar{V_i})$, these two definitions are equivalent. Therefore, normalized cut can be viewed as a special case of conductance.

X is the module assignment matrix and h_i presents the ith column of X. X is in the following feasible solution space

$$\mathcal{F}_k = \{X : X\mathbf{1}_k = \mathbf{1}_n, X_{ij} \in \{0, 1\}\}. \tag{2.14}$$

$\mathbf{1}_k$ and $\mathbf{1}_n$ are all one vectors with dimension k and n respectively.

Detecting k normalized cuts can be formulated as

$$\begin{aligned}\sum_i N_{cut}(V_i) &= \sum_i \frac{|E(V_i,\bar{V_i})|}{\mathrm{vol}(V_i)} \\ &= \sum_i \frac{x_i^T(D-A)x_i}{x_i^T D x_i} \\ &= \mathrm{Tr}\left(\begin{bmatrix} \frac{x_1^T(D-A)x_1}{x_1^T D x_1} & 0 & 0 \\ 0 & \frac{x_2^T(D-A)x_2}{x_2^T D x_2} & 0 \\ 0 & 0 & ... \end{bmatrix}_{k\times k}\right) \\ &= \mathrm{Tr}\left[(X^T D X)^{-1} X^T (D-A) X\right]\end{aligned} \tag{2.15}$$

Our goal is to find the optimal solution X that can minimize the k-way normalized cuts. The problem can be casted into the following optimization problem:

$$\begin{aligned}\min: &\quad \mathrm{Tr}\left[(X^T D X)^{-1} X^T (D-A) X\right] \\ s.t. &\quad X \in \mathcal{F}_k.\end{aligned} \tag{2.16}$$

Furthermore, we find that the above problem can be further simplified to the following equivalent problem

$$\begin{aligned}\max: &\quad \mathrm{Tr}\left[(X^T D X)^{-1} X^T (A) X\right] \\ s.t. &\quad X \in \mathcal{F}_k.\end{aligned} \tag{2.17}$$

Defining $S = \mathrm{Diag}(s_1, s_2, ..., s_k) = (X^T D X)^{1/2}$, $Y = D^{1/2} X S^{-1}$ and $W = D^{-1/2} A D^{-1/2}$, we finally have

$$\begin{aligned} \max \quad & \mathrm{Tr}(Y^T W Y) \\ s.t. \quad & Y^T Y = I_k, \\ & (D^{-1/2} y_j)_i \in \{0, s_j^{-1}\}, \forall i, j, \\ & Y S \mathbf{1}_k = \mathrm{diag}(D^{-1/2}), \\ & S = \mathrm{Diag}(s_1, s_2, ..., s_k) \in R_+^n. \end{aligned} \tag{2.18}$$

The above problem is a NP-hard combinatorial optimization problem.

Spectral method

A spectral method can be used to approximate the solution of (2.20). Consider the following relaxed problem of (2.20)

$$\begin{aligned} \max \quad & \mathrm{Tr}(Y^T W Y) \\ s.t. \quad & Y^T Y = I_k. \end{aligned} \tag{2.19}$$

Although $\mathrm{Tr}(Y^T W Y)$ is not convex because W may not be positive semi-definite matrix, we can retrieve the optimal solution supported by Kay Fan theorem. Based on the theorem, the optimal Y^* is attained at $Y^* = U_k$, whose columns are the eigenvectors corresponding to the k largest eigenvalues of W. k-means clustering can be used to obtain a feasible solution in the original constraint set.

Ky Fan Theorem: *Let T be a symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ and the corresponding eigenvectors $U = (u_1, ..., u_n)$. Then*

$$\sum_{i=1}^{k} \lambda_i = \max_{X^T X = I_k} \quad \mathrm{Tr}(X^T T X) \tag{2.20}$$

Moreover, the optimal X^* is given by $X^* = [u_1, ..., u_k]Q$ with Q being an arbitrary orthogonal matrix.

Other methods

The k normalized cut problem (2.13) can also be solved by semi-definite programming [95]. For finding the exact solution, one can reformulate (2.13) into a mixture integer programming and solve it by using linear programming toolbox [28].

2.1.2.2 Local algorithm based on personalized PageRank vector

Andersen [6] developed a local algorithm, called PageRank-Nibble, to find a low-conductance set near a specific starting node in the network based on a personalized PageRank vector. The conductance of the set C identified by the algorithm is at most $f(\phi(C))$, where $f(\phi(C))$ is $\Omega(\frac{\phi(C)^2}{\log m})$ (m: the number of edges in G). Furthermore, the local algorithm can find C in time $O(\frac{m\log^4 m}{\phi(C)^3})$. PageRank-Nibble provides us a powerful weapon to find a low-conductance set near a specific node with theoretical guarantee with only linear computational complexity with respect to $|C|$.

```
Algorithm: ApproximatePageRank ($i$, $\alpha$, $\xi$)
    Let $p = \mathbf{0}$ and $r = e_i$.
    While $\max_{i \in V} \frac{r(i)}{\deg(i)} \geq \xi$
        Choose a node $i$ with $\frac{r(i)}{\deg(i)} \geq \xi$.
        Apply push($i$, $p$, $r$) and update pr, $r$.
    Return $p \approx \mathrm{pr}(\alpha, e_i)$.
```
Figure 2.1: The algorithm to approximate the personalized PageRank vector.

Approximation of a personalized PageRank vector

One fundamental step of PageRank-Nibble is to approximate the personalized PageRank vector around node i. Following [6], the lazy variation of PageRank vector is defined as

$$\mathrm{pr}(\alpha, s) = \alpha s + (1-\alpha)\mathrm{pr}(\alpha, s)\mathcal{P}, \tag{2.21}$$

where α is a constant in $(0, 1]$ called the teleportation constant, s is a distribution called preference vector and $\mathcal{P} = \frac{1}{2}(I - D^{-1}A)$ is the transition probability matrix of the lazy random walk on G. The personalized PageRank vector used in (2.21) requires $s = e_i$, where e_i is an all zeros vector with one on the ith entry, meaning that the preference vector concentrates on the ith node. The pseudo code of the approximation is displayed in Fig. 2.1, where the subroutine used is shown in Fig. 2.2. The technical and theoretical details can be found in [6]. The approximation algorithm has the proven guarantee: $\max_{i \in V} \frac{r(i)}{\deg(i)} \geq \xi$.

```
Algorithm:  push ($p$, $r$)
    Let $p' = p$ and $r' = r$.
      $p'(i) = p(i) + \alpha r(i)$.
      $r'(i) = (1-\alpha)\frac{r(i)}{2}$.
      For node $j$ ($A_{ij} = 1$), $r'(j) = r(j) + (1-\alpha)\frac{r(i)}{2\deg(i)}$.
    Return $p'$ and $r'$.
```

Figure 2.2: The push algorithm.

PageRank-Nibble

Once we obtain the approximation of personalized PageRank vector p near node i, then we can sort the nodes around i base on $\frac{p(j)}{\deg(j)}$. Assuming $v_1, v_2, ..., v_{Np}$ is the ordering of the nodes around i such that $\frac{p(v_i)}{\deg(v_i)} \geq \frac{p(v_{i+1})}{\deg(v_{i+1})}$, we compute the conductance of the set $C_j = \{v_1, v_2, ..., v_j\},\ j \in \{0, 1, ..., N_p\}$. The set C^* with the smallest conductance $C^* = arg \min_{C_j} \phi(C_j)$ is the result produced by the PageRank Nibble algorithm.

2.1.2.3 Markov clustering algorithm (MCL)

MCL [26] is a graph clustering algorithm based on manipulation of transition probabilities between nodes of the graph. The underlying principle of MCL is still unknown, which has not prevented its successes on many real-world applications. We category MCL as a algorithm related to finding low-conductance sets in network because its similarity to Nibble [100]. Nibble is a local clustering algorithm to recover the low-conductance set C around node i, whose running time is almost-linear with respect to $|C|$. Actually, the PageRank-Nibble algorithm is a variation of Nibble.

MCL iteratively implements “Expand” and “Inflation” operations on the transition matrix $\mathcal{P}$ of the underlying Markov chain of random walk on the given network G. In the “Expand” step, we perform $\mathcal{P}_t = \mathcal{P}_{t-1}\mathcal{P}_{t-1}$. “Inflation” operation follows to compute $\mathcal{P}_t(i,j) = \dfrac{\mathcal{P}_t^r(i,j)}{\sum_{i=1}^n \mathcal{P}_t^r(i,j)}$. Those two operations iterate until $\mathcal{P}_t$ converges. Each row of $\mathcal{P}_t$ contains the membership information corresponding to one cluster. Generally, most of the rows of $\mathcal{P}_t$ converge to all zero vectors. The only parameter of MCL is r, which controls the size of the modules in average sense. If r is large, then the module size tends to become small.

In comparison to MCL, Nibble also consists of two major steps, which is the random walk propagation and the removal of unrelated nodes. Nibble computes the random walk probability vector within the first several steps. It starts with vector $q_0 = e_v$. In the random walk propagation step, $q_t = \mathcal{P}q_{t-1}$ is performed. In the removal step, the nodes with probabilities smaller than $\varepsilon \cdot \deg(i)$, where $\varepsilon = \phi^2/(\log^3 m2^b)$, are zero out.

Comparing MCL with Nibble, intuitively, both of them have two similar steps, random walk propagation and node deletion. The similarity may explain why MCL yields reasonable results.

2.1.3 Community detection based on non-negative matrix factorization

Recently, NMF has been successfully applied to network partition problem [47, 24, 106, 19]. The authors in [24, 47] proposed to decompose the adjacency matrix of a network with undirected edges into symmetric non-negative components to identify communities under the assumption that all modules consist of highly connected nodes. Further investigation has demonstrated its potential for detecting overlapping modules in networks [24, 76, 120].

The authors in [47, 24] propose to decompose the corresponding adjacency matrix A for a given network into symmetric components for community detection:

$$\min_{X \geq 0}: \Gamma(X) = \left\| A - XX^T \right\|_F^2, \tag{2.22}$$

where X is a non-negative matrix of size $n \times k$ and k is the number of potential modules. X can be naturally interpreted as the module assignment matrix. A multiplicative updating algorithm SymNMF_MU [24] has been proposed to solve this problem (2.22).

$$X_{ik} \leftarrow X_{ik} \left(1 - \gamma + \gamma \frac{(AX)_{ik}}{(XX^TX)_{ik}} \right), \gamma \in (0, 1] \tag{2.23}$$

However, SymNMF_MU may not converge to a stationary point. SymNMF_NT [47] is a Newton-like algorithm, which solves the problem (3.17) by lining up the columns of X. SymNMF_NT converges to a stationary point. However, it has relatively larger memory consumption requirement [47].

2.2 Module identification based on interaction patterns

There are many algorithms developed to identify functional modules with the consideration of interaction patterns. For example, Power Graph (PG) [86] greedily collects topological similar nodes into the same module based on Jaccard Index similarity. Graph Summarization (GS) [67, 66] uses the minimum description length principle to group nodes with similar interaction patterns. However, both PG and GS are solved by greedy algorithms, which can not guarantee the global optimality. Additionally, they tend to over-segment the network to get relatively small modules based on our empirical experience. A Bayesian framework [35] based on a stochastic block modeling formulation has been developed to identify modules as well as the optimal number of modules. However, the algorithm only guarantees to converge to local optima. Reichardt [84] has proposed to solve block modeling module identification by optimally mapping the given network to an image graph using simulated annealing (SA), and several optimization strategies also have been proposed to accelerate the original SA algorithm [107, 108]. NMF based optimization frameworks [106, 19] are proposed to find functional modules by explicitly considering the underlying image graph of a network, however, there is no convergence guarantee for the developed algorithms. In this section, we review the optimization formulations of block modeling.

2.2.1 Block modeling based on the image graph

We first review block modeling module identification by functional role decomposition proposed by [85, 83, 80]. For module identification of network G, we search for a non-overlaping module mapping τ which assigns n nodes in V to q different modules: $\tau : V \mapsto U$, in which $U = \{u_1, \ldots, u_q\}$ represent the module space—a set of virtual module nodes (Fig. 2.3). To obtain the optimal module identification,

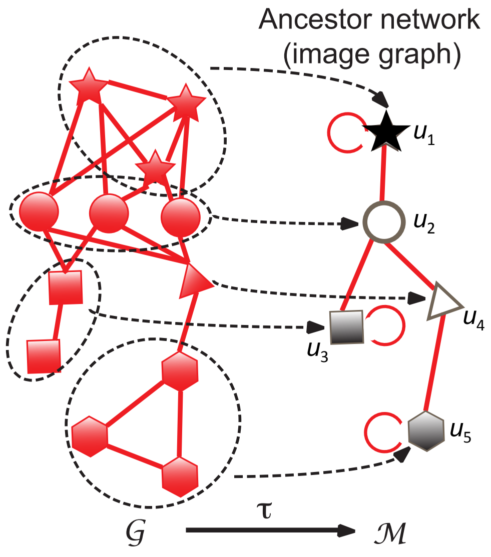


Figure 2.3: Mapping to the module space as an introduced image graph. The shaded nodes denote highly connected modules and the hollow nodes are for modules of which nodes have similar interaction patterns.

[85, 83, 80] introduce the virtual image graph $M = \{U, I\}$ in the module space to preserves the original network interactions by the edges I. Note that when $q = n$, the obvious optimal image graph is the network itself.

Mathematically, the optimal τ should minimize the mismatch between the given network G and the introduced image graph M. Suppose we have the adjacency matrix of the image graph by B, where B_{rs} records the interaction between module nodes u_r and u_s. In order to make the image graph $B_{\tau_i \tau_j}$ match the network A_{ij} as much as possible, we search for the optimal module mapping τ and minimize the following error function [83, 80]:

$$E(\tau, B) \quad = \quad \frac{1}{M} \sum_{i \neq j}^{N} (A_{ij} - B_{\tau_i \tau_j})(w_{ij} - p_{ij}),$$

in which w_{ij} denotes the weight given to the edge between node v_i and v_j (in this paper $w_{ij} = 1$ when node v_i and v_j have an interaction, and $w_{ij} = 0$ otherwise); $M = \sum_{i \neq j}^{N} w_{ij}$ is used to bound the error function between 0 and 1; $(w_{ij} - p_{ij})$ denotes the error made on the edge between v_i and v_j with p_{ij} as the penalty for the mismatches of the corresponding absent edges. Self-links in the original network are not considered with both $w_{ii} = 0$ and $p_{ii} = 0$. Typically, p_{ij} is chosen to make the total mismatches error on existing edges ($w_{ii} > 0$) equal that on absent edges ($w_{ii} = 0$): $\sum_{i \neq j}^{N} A_{ij}(w_{ij} - p_{ij}) = \sum_{i \neq j}^{N}(1 - A_{ij})p_{ij}$. In order to guarantee this equality, we follow one of possible choices [83] to let $p_{ij} = \frac{\sum_{k \neq i} w_{ik} \sum_{l \neq j} w_{lj}}{\sum_{k \neq l} w_{kl}}$.

From the equation of $E(\tau, B)$, we find that $B_{\tau_i \tau_j} = A_{ij}$, which means the image graph preserves an edge (either existing edge or absent edge) in the original network, leads to $E(\tau, B) = 0$. Otherwise, $B_{\tau_i \tau_j} \neq A_{ij}$, which means image graph does not preserve an edge in original network, leads to $E(\tau, B) = p_{ij}$ when miss-matching the absent edge or $E(\tau, B) = w_{ij} - p_{ij}$ when miss-matching the existing edge. By further investigating $E(\tau, B)$, we find that minimizing $E(\tau, B)$ is equivalent to maximize $\frac{1}{M} \sum_{i \neq j}^{N}(w_{ij} - p_{ij}) B_{\tau_i \tau_j}$ which can be rewritten as $\max_{\tau, B} \frac{1}{2M} \sum_{i \neq j}^{N}(w_{ij} - p_{ij})(2B_{\tau_i \tau_j} - 1)$ by using binary trick. Furthermore, we can formulate the objective function as $Q(\tau, B) = \frac{1}{2M} \sum_{r,s} \sum_{i \neq j}^{N}(w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s}(2B_{rs} - 1)$. For the original nodes assigned to module node u_r and u_s by τ, we have the corresponding term as $\sum_{i \neq j}^{N}(w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s}(2B_{rs} - 1)$, in which $\delta_{\tau_i r}$ is the indicator function that takes 1 when $\tau_i = r$ and 0 otherwise. It is clear that the optimal solution for B_{rs} with a given τ is to set $B_{rs} = 1$ when its corresponding term $\sum_{i \neq j}^{N}(w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s}$ is larger than 0, and 0 otherwise. Hence, the optimal solutions of τ and B are naturally decomposed. The optimal image graph B can be derived in a straightforward way once we have the optimal module mapping τ, which maximizes the following equivalent objective function:

$$Q^*(\tau) = \frac{1}{2M}\sum_{r,s}^{q}\left|\sum_{i\neq j}^{N}(w_{ij}-p_{ij})\delta_{\tau_i r}\delta_{\tau_j s}\right|. \tag{2.24}$$

The maximization problem (2.24) is NP hard as it can be polynomially transformed to the classical quadratic assignment problem [61]. In [80], SA has been applied for the optimization, in which the time complexity increases quadratically with increasing q. While in practice, the search space for annealing parameters gets larger with increasing q too, and it can obtain high quality results only for very slow cooling schedules. For a large q (≥ 100), SA will be a time-consuming procedure.

2.2.2 Block modeling based on NMF

For block modeling module identification, one recent algorithm—BNMF [19]—has been derived base on the following formulation:

$$\min_{X\geq 0,\ 0\leq M\leq 1}: \left\|A - XMX^T\right\|_F^2 + \lambda\left\|M^{ideal} - M\right\|_F^2, \tag{2.25}$$

where M and M^{ideal} represent the adjacency matrices of the introduced image graph and the "ideal image matrix", respectively. M^{ideal} is the function of M, which is defined by $M_{ij}^{ideal} = \underset{u\in\{0,1\}}{argmin}\,|u - M_{ij}|$ and approximated by a sigmoid function in the proposed projected descent algorithm. However, there is no convergence proof provided for BNMF.

2.3 Module identification for multiple networks

It is well known that protein interaction networks are noisy. There exist many false positive and false negative edges. Therefore, it is very challenging to assign functional roles to proteins and separate true protein-protein interactions from false positive interactions. Across species comparison may provide us a valuable framework to address those challenges. Finding conserved modules in biological networks of multiple species has attracted researchers attention due to its capacity of identifying network regions that conserved in their sequences and interaction patterns across species. Algorithms of pairwise and multiple networks [89, 54, 99, 46, 22, 94, 39, 116] have been successfully applied for searching conserved pathways and complexes. Align-Nemo [22], NetworkBlast [94] and MaWISh [46] can only handle pairwise networks by constructing pairwise networks into alignment networks. NetworkBlast-M [39] and OrthoClust [116] can deal with multiple networks but suffer from the low coverage and the resolution problems [30], respectively.

The most challenging part of finding conserved modules across networks, again an NP hard problem, is how to compromise between the time and space complexities of the algorithms and the accuracy of the alignment results. One fundamental problem is the data representation of multiple networks. Based on different data representations, different algorithms are designed to solve the difficult tasks. Therefore, we go through the state-of-the-art algorithms by the ways they integrate the multiple networks.

In this section, a set of K networks $\mathcal{G} = \{G_1(V_1, E_1), G_2(V_2, E_2), ..., G_K(V_K, E_K)\}$ is presented by a set of adjacency matrices $\mathcal{A} = \{A_1, A_2, ..., A_K\}$, and the orthology relationships between them are kept in S, where $S(i, j), \forall i \in G_s, j \in G_t$ is the orthology correspondence between node i in network G_s and node j in network G_t.

For algorithms using pairwise networks, a pair of networks $\{G_1, G_2\}$ and their orthology relationship S_{12} are represented by a product graph, which is the Cartesian product of the two networks containing every possible combinations between nodes across species. [54, 99] are developed based on the product graphs. An alignment network consists of nodes across networks with orthology correspondence and edges, which are the conserved interactions. The alignment network is the basic data representation for many state-of-the-art pairwise networks alignment algorithms [99, 46, 22, 94]. Researchers extend pairwise networks alignment to multiple networks alignment by representing the integration of $\mathcal{G}$ and S using a K layer graph [39, 116], which is also called layered alignment graph $G_H(\cup_i V_i, E)$ where E is the union of $\cup_i E_i$ and E_H denoting inter-layer edges corresponding to S. The existing multiple networks alignment algorithms [39, 116] are based on K layered alignment graphs due to computational considerations.

2.3.1 Algorithms based on the product graph

Let G_a and G_b be two biological networks to align. Two networks has N_a and N_b nodes respectively. We define $B \in \mathbf{R}^{(N_a \times N_b)\times(N_a \times N_b)}$ as the Cartesian product graph G_B from G_a and G_b: $B = G_a \otimes G_b$. Denote the all one vector $\mathbf{1} \in \mathbf{R}^{N_a \times N_b}$ and

$$\bar{B} = B \times \text{Diag}(B\mathbf{1})^{-1}, \tag{2.26}$$

where $\text{Diag}(B\mathbf{1})$ can be considered as a degree matrix with $B\mathbf{1}$ on its diagonal and all the other entries equal zero. $\bar{B}$ is the transition probabilities for the underlying Markov random walk in IsoRank [99]. It is well known that if G_a and G_b are connected networks and neither of them is bipartite graph, then the corresponding Markov chain represented by $\bar{B}$ is irreducible and ergodic, and there exists a unique stationary distribution for the underlying state transition probability matrix $\bar{B}$.

2.3.1.1 IsoRank

[99] IsoRank is a pairwise network alignment algorithm based on random walk on the product network G_B. IsoRank is derived based on the intuition that a pair of nodes is aligned together if their neighborhood nodes aligned together. Instead of finding the alignment directly, IsoRank first computes the similarities between all

node pairs in two networks based on their neighborhood topologies and sequence information. The all-to-all similarity scores can be obtained by finding a right maximal eigenvector of the matrix $\bar{B}$: $\bar{B}\mathbf{x} = \mathbf{x}$ and $\mathbf{1}^T\mathbf{x} = 1,\ \mathbf{x} \geq 0$. When two networks are of reasonable size, spectral methods as well as power methods can be implemented to solve the eigen-system. When we obtain r, a bipartite network on networks G_a and G_b can be constructed, edges of which are the similarities scores stored in r. The alignment result of IsoRank then can be attained by solving the maximal matching problem in the bipartite network.

2.3.1.2 IsoRankN

[54] IsoRankN is the extension of IsoRank. Computationally, it is very hard to store the product network of three or more networks. For example, for three 100-node networks, the size of the product network of them is 1000,000$\times$ 1000,000. The size of real-world networks are much larger than 100, therefore it is memory prohibitive to use the product network for multiple networks alignment problem. Instead, IsoRankN computes the pairwise alignment similarity scores and then constructs a k-partite network. Then the local algorithm based on the personalize PageRank vector introduced in section 2.1.2 is applied to the k-partite network to find the alignment results.

2.3.2 Algorithms based on the alignment network

Another widely used data representation is the alignment network. For two networks $G_a(V_a, E_a)$ and $G_b(V_b, E_b)$ and their across-network correspondence S, the alignment network $\bar{G}$ can be constructed as following. For nodes u in G_a and v in G_b, if $S(u, v) > 0$, then the node pair (u, v) is considered as a node in $\bar{G}$. If $(u, u') \in E_a,\ (v, v') \in E_b$ and $S(u, v) > 0,\ S(u', v') > 0$, then there is an edge between node (u, v) and (u', v') in $\bar{G}$.

2.3.2.1 AlignNemo

[22] AlignNemo propose a heuristic way to build a union network, which is a variant alignment network. A seed-expansion algorithm is then implemented to identify the conserved complexes in two protein interaction networks.

2.3.2.2 MaWISh

[46] MaWISh, which is short for Maximum Weight Induced Subgraph, is a framework for alignment of pairwise protein interaction networks with the consideration of the influence of evolution. Based on the duplication/divergence models, a classic evolution model, MaWISh converts the match, mismatch and duplication of both sequence and network structure into mathematical formulation. Then a heuristic algorithm is developed to find the alignment on the alignment network.

2.3.2.3 NetworkBlast

[94] NetworkBlast is also developed based on the alignment network. After the construction of a alignment network, d-subnetworks are identified if the densities of them are statistically significant. NetworkBlast then follows the seed-expansion algorithm.

2.3.3 Algorithms based on the layered alignment network

2.3.3.1 NetworkBlastM

[39] NetworkBlastM is multiple networks version of NetworkBlast. Because construction of the alignment networks of multiple networks is limited by the memory capacity, NetworkBlastM finds d-subnetworks on the layered alignment network instead. NetworkBlastM and NetworkBlast are almost the same except their data representation of the multiple networks.

2.3.3.2 OrthoClust

[116] OrthoClust is a computational framework to identify conserved cross-species modules by integrating the co-association networks of individual species with the orthology relationships between species. Recently, it has been successfully applied to the comparative analysis of the transcriptome across distant species (human, fly and worm) [32].

For K networks of different species with orthology relationship, the objective function of OrthoClust for identifying k conserved modules across species is as following

$$\mathcal{H} = \sum_i X_i^T (A_i - P_i) X_i + \kappa \hat{X}^T S \hat{X}, \tag{2.27}$$

where X_i is the module assignment matrix of size $|V_i| \times k$ for network G_i and $\hat{X}$ is the stack of all module assignment matrix

$$\hat{X}^T = \left[X_1^T, X_2^T, ..., X_k^T \right]. \tag{2.28}$$

The first term of (2.27) is the sum of the modularities of every networks, which aims to make sure modules in each network are as cohesive as possible. The second term of (2.27) computes the conserved orthology information for the current module assignment results. Therefore, maximization of (2.27) leads us to find conserved modules, which have densely connected structures as well as close orthology correspondence.

In [116], the authors propose a simulated annealing algorithm to solve the combinatorial optimization. To obtain a stable solution, the simulated annealing algorithm is implemented many times to yield many solutions and the overlapped parts of those solutions are considered to be the final output of OrthoClust. The procedure of OrthoClust implies that OrthoClust may be very time consuming and the quality of the solution is hard to guarantee. Actually, in their open source codes, published online, we find they use a greedy algorithm based on the framework of [116]. Because the framework of OrthoClust is based on network modularity [116], it inevitably suffers from the resolution problem, which force it to ignore functional modules with small sizes.

3. BLOCK MODELING FOR INDIVIDUAL PROTEIN INTERACTION NETWORKS*

In this chapter, we introduce three algorithms we developed based on the concept of block modeling and verify their performance on synthetic and real-world networks. First, in section 3.1, we present an efficient algorithm to solve the classic block modeling formulation [80], which is originally solved by simulated annealing algorithm. Then, to overcome the resolution problem of the block modeling formulation [80], we propose a novel formulation based on two-hop random walk on networks and develop algorithms to identify non-overlapping and overlapping functional modules in section 3.2. At last, in section 3.3, we introduce an algorithm for a flexible NMF based formulation with sparse regularization to detect functional modules based on interaction patterns with convergence guarantee.

3.1 Sub-gradient Frank-Wolfe method with path generation

Functional module identification based on block modeling is NP hard. The optimization formulation in [80] has an objective function that is highly nonlinear and non-convex with many local optima. It is computationally prohibitive to obtain the optimal modules in large-scale networks. Simulated Annealing (SA) has been proposed to obtain the global optimum in [80]. However, it requires a very slow cooling down procedure to guarantee the solution quality. In addition, its computational time increases quadratically with the number of modules to identify. In order to

*Part of the content of the first section in this chapter is reprinted with permission from "A novel subgradient-based optimization algorithm for blockmodel functional module identification" by Yijie Wang and Xiaoning Qian, BMC Bioinformatics, 14(Suppl 2): S23, 2013, Copyright 2013 by BMC Bioinformatics. Part of the content of the second section in this chapter is reprinted with permission from "Functional module identification in protein interaction networks by interaction patterns" by Yijie Wang and Xiaoning Qian, Bioinformatics, 30(1): 81-93, 2014, Copyright 2014 by Bioinformatics.

identify fine-grained functional modules in genome-scale biological networks, more efficient algorithms are needed.

In this section, we propose an efficient optimization method—subgradient with path generation(SGPG) method to solve the difficult non-convex combinatorial optimization problem by combining the subgradient(SG) convex programming with heuristic path generation (PG) method. PG is proposed to make use of obtained local optima to search for better solutions. We evaluate the performances of our functional module identification algorithms with SA in two biological networks: *Saccharomyces cerevisia* (*Sce*) PPI network from the Database of Interacting Proteins (DIP) [90] and *Homo sapien* (*Hsa*) PPI network collected from the Human Protein Reference Database (HPRD version 9) [82]. The results show that our new SGPG method achieves high quality solutions with significantly reduced computation time compared to SA. Furthermore, we have implemented SGPG and the Markov Clustering (MCL) algorithm [26] to these two PPI networks. The results demonstrate that SGPG can identify additional biologically meaningful modules that MCL may miss, which may lead to a better understanding of functional organization of these biological networks.

3.1.1 Methodology

3.1.1.1 Block modeling framework

We first review block module identification by functional role decomposition proposed by [85, 83, 80]. Given a biological network, we can mathematically represent it as a graph $G = \{V, E\}$, where $V = \{v_1, v_2, \ldots, v_n\}$ are network nodes representing biomolecules, n is the number of nodes in G, and E are edges representing interactions among molecules. The network topology can be completely determined by its corresponding adjacency matrix A, where matrix entries A_{ij} record the interactions between the pairs of nodes: v_i and v_j in G. For module identification, we search for a non-overlap module mapping τ which assigns n nodes in V to q different modules: $\tau : V \mapsto U$, in which $U = \{u_1, \ldots, u_q\}$ represent the module space—a set of virtual module nodes (Fig. 2.3). To obtain the optimal module identification, [85, 83, 80] introduced the virtual *image graph* $\mathcal{M} = \{U, I\}$ in the module space to preserves the original network interactions by the edges I. Note that when $q = n$, the obvious optimal image graph is the network itself.

Mathematically, the optimal τ should minimize the mismatch between the given network G and the introduced image graph $\mathcal{M}$. Suppose we have the adjacency matrix of the image graph by B, where B_{rs} records the interaction between module nodes u_r and u_s. In order to make the image graph $B_{\tau_i\tau_j}$ match the network A_{ij} as much as possible, we search for the optimal module mapping τ and minimize the following error function [83, 80]:

$$E(\tau, B) = \frac{1}{M}\sum_{i\neq j}^{n}(A_{ij} - B_{\tau_i\tau_j})(w_{ij} - p_{ij}),$$

in which w_{ij} denotes the weight given to the edge between node v_i and v_j (in this section $w_{ij} = 1$ when node v_i and v_j have an interaction, and $w_{ij} = 0$ otherwise); $M = \sum_{i\neq j}^{n} w_{ij}$ is used to bound the error function between 0 and 1; $(w_{ij} - p_{ij})$ denotes the error made on the edge between v_i and v_j with p_{ij} as the penalty for the mismatch of the corresponding absent edges. Self-links in the original network are not considered with both $w_{ii} = 0$ and $p_{ii} = 0$. Typically, p_{ij} is chosen to make the total mismatch error on existing edges ($w_{ii} > 0$) equal to that on absent edges ($w_{ii} = 0$): $\sum_{i\neq j}^{n} A_{ij}(w_{ij} - p_{ij}) = \sum_{i\neq j}^{n}(1 - A_{ij})p_{ij}$. In order to guarantee this equality, we follow one of possible choices [83] to let $p_{ij} = \frac{\sum_{k \neq i} w_{ik} \sum_{l \neq j} w_{lj}}{\sum_{k \neq l} w_{kl}}$.

From equation of $E(\tau, B)$, we find that $B_{\tau_i \tau_j} = A_{ij}$, which means image graph preserves an edge (either existing edge or absent edge) in the original network, leads to $E(\tau, B) = 0$. Otherwise, $B_{\tau_i \tau_j} \neq A_{ij}$, which means image graph does not preserve an edge in original network, leads to $E(\tau, B) = p_{ij}$ when miss-matching the absent edge or $E(\tau, B) = w_{ij} - p_{ij}$ when miss-matching the existing edge. By further investigating $E(\tau, B)$, we find that minimizing $E(\tau, B)$ is equivalent to maximize $\frac{1}{M} \sum_{i \neq j}^{n} (w_{ij} - p_{ij}) B_{\tau_i \tau_j}$ which can be rewritten as $\max_{\tau, B} \frac{1}{2M} \sum_{i \neq j}^{n} (w_{ij} - p_{ij})(2B_{\tau_i \tau_j} - 1)$ by using binary trick. Furthermore, we can formulate the objective function as $Q(\tau, B) = \frac{1}{2M} \sum_{r,s} \sum_{i \neq j}^{n} (w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s} (2B_{rs} - 1)$. For the original nodes assigned to module node u_r and u_s by τ, we have the corresponding term as $\sum_{i \neq j}^{n} (w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s} (2B_{rs} - 1)$, in which $\delta_{\tau_i r}$ is the indicator function that takes 1 when $\tau_i = r$ and 0 otherwise. It is clear that the optimal solution for B_{rs} with a given τ is to set $B_{rs} = 1$ when its corresponding term $\sum_{i \neq j}^{n} (w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s}$ is larger than 0, and 0 otherwise. Hence, the optimal solutions of τ and B are naturally decomposed. The optimal image graph B can be derived in a straightforward way once we have the optimal module mapping τ, which maximizes the following equivalent objective function:

$$Q^*(\tau) = \frac{1}{2M} \sum_{r,s}^{q} \left| \sum_{i \neq j}^{n} (w_{ij} - p_{ij}) \delta_{\tau_i r} \delta_{\tau_j s} \right|. \tag{3.1}$$

The maximization problem (3.1) is NP hard as it can be polynomially transformed to the classical quadratic assignment problem [61]. In [80], SA has been applied for the optimization, in which the time complexity increases quadratically with increasing q. While in practice, the search space for annealing parameters gets larger with increasing q too, and it can obtain high quality results only for very slow cooling schedules. For a large q (≥ 100), SA will be a time-consuming procedure.

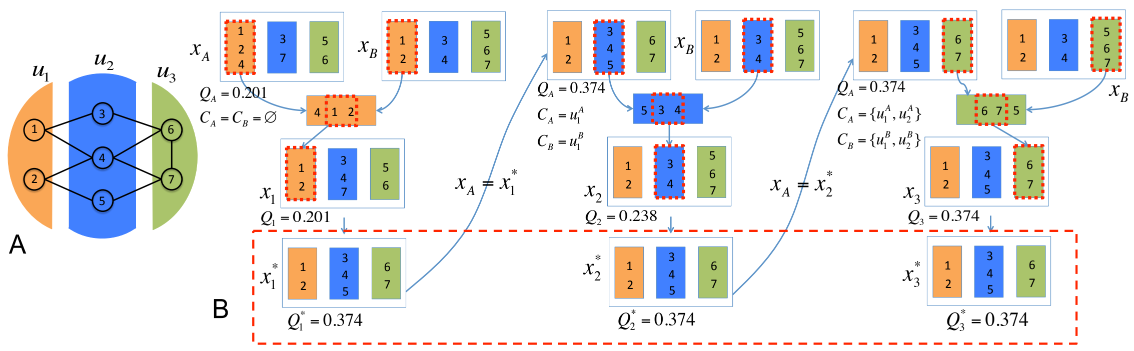


Figure 3.1: An example of path generation: A. Network structure. B. Path generation procedure.

3.1.1.2 Subgradient with path generation method (SGPG)

We propose to speed up the block module identification problem by convex programming and heuristic path generation method. PG is originally proposed in this section as a new useful heuristic to combine with subgradient algorithms to efficiently solve the hard combinatorial optimization problem. The combination of SG and PG can dramatically reduce the computational time with competitive numerically and biological performance comparing to SA method for the block module identification problem. The basic idea is first using the fast sungradient convex programming method to obtain the local optima, then using path generation method to search for better solutions. SG and PG are described in the following sections.

3.1.1.3 Subgradient convex programming method (SG)

Module identification problem in matrix form

We now reformulate the module identification problem in (3.1) in matrix form by introducing an assignment matrix X corresponding to the module mapping τ. To identify q non-overlapping functional modules in G, the assignment matrix X is defined as an $n \times q$ matrix with each entry $X_{ir} = 1$ when v_i is assigned to the module u_r or equivalently, $\tau_i = r$; and $X_{ir} = 0$ otherwise. In other words, $X_{ir} = \delta_{\tau_i r}$.

Each column in X corresponds to an image module node in which all the assigned network nodes take the value "1". We further use W to denote the weight matrix with each entry as the corresponding edge weight w_{ij}, and P as the penalty matrix with each entry as the corresponding penalty p_{ij}. The objective function in (3.1) can be rewritten in the following equivalent matrix form:

$$Q^{*}(\tau) = f(X) = \left\| X^{T} \left(W - P\right) X \right\|_{L_1} \tag{3.2}$$

The sum of each row in X has to be the unity and the columns of X are orthogonal to each other. In addition, if we assume that each node has to be assigned to one module, the assignment matrix X has to satisfy the normalization condition $X1_q = 1_n$, in which 1_q and 1_N denote the q-dimensional and n-dimensional vectors of all ones. Hence, the optimal solution for the assignment matrix X lie in the space $\phi = \{X \in \{0,1\}^{n\times q},\ X1_q = 1_n\}$, we have the convex programming formulation based on image graph:

$$\begin{aligned} &\min_X :\ F(X) := -\left\| X^{T} \left(W - P\right) X \right\|_{L_1} \\ &s.t.\ \ X \in \phi. \end{aligned} \tag{3.3}$$

Note that we have converted our maximization problem into a minimization problem for the convenience of introducing subgradient methods in convex programming [11]. We denote $Q = X^{T}\left(W - P\right)X$ with its entries $Q_{rs} = X_r^{T}\left(W - P\right)X_s$, where X_r is the rth column of X. Again, with the optimal assignment matrix X, we can derive the topology of the image graph B: $B_{rs} = 1$ if $Q_{rs} > 0$, and 0 otherwise.

Subgradient

We note that the optimization problem (3.3) is a non-smooth combinatorial optimization problem as the objective function involves the L_1 norm of the matrix

Q. To solve this hard optimization problem, we first relax the binary constraints $X \in \{0,1\}^{n\times q}$ in (3.3) by continuous relaxation $X \in [0,1]^{n\times q}$ and use γ to represent the relaxed constraint set, which is a convex hull after relaxation. To address the nonlinearity of the matrix L_1 norm objective function $F(X) = -\left\|Q\right\|_{L_1}$ with the relaxed linear constraints, we propose to use Frank-Wolfe algorithm [11] to iteratively solve the following optimization problem with a linear objective function from the approximation by the first-order Taylor expansion:

$$\begin{aligned} &\min_X : F(X^t) + < \nabla F(X^t), (X - X^t) > \\ &s.t. \;\; X \in \gamma, \end{aligned} \tag{3.4}$$

where X^t is the current solution, $<,>$ is the inner product operator, and the new objective function is from the first-order Taylor expansion. The problem (3.4) at each iteration is a linear programming problem to search for the local extreme point along the gradient $\nabla F(X^t)$ as in steepest descent. However, as previously stated, $F(X^t)$ takes the matrix L_1 norm, which is non-smooth, and therefore non-differentiable. To address this last complexity, we apply subgradient methods [11] to replace $\nabla F(X^t)$ by a subgradient $\partial F(X^t)$ instead [8]:
Definition (Subgradient): A matrix $\partial F \in \mathcal{R}^{m\times n}$ is a subgradient of a function $F : \mathcal{R}^{m\times n} \to R$ at the matrix $\bar{X} \in \mathcal{R}^{m\times n}$ if $F(Z) \geq F(\bar{X}) + < \partial F, (Z - \bar{X}) >, \forall Z \in \mathcal{R}^{m\times n}$.

In our case, the subgradient of the matrix L_1 norm can be presented by its dual norm—matrix L_∞ norm, which is used to derive the subgradient $\partial F(X^t)$. Similar to the derivation for the subgradient of the L_1 norm of vectors in L_1 regularization in [8], we show that the subgradient of the L_1 norm of any matrix $\bar{X}$ is

$$\partial \left\|\bar{X}\right\|_{L_1} = \begin{cases} \left\{Y \in \mathbf{R}^{m\times n}; \|Y\|_{L_\infty} \le 1)\right\}, & \text{if } \bar{X} = \mathbf{0}; \\ \left\{Y \in \mathbf{R}^{m\times n}; \|Y\|_{L_\infty} \le 1 \textit{ and } < Y, \bar{X} >= \left\|\bar{X}\right\|_{L_1}\right\}, & \text{otherwise,} \end{cases} \tag{3.5}$$

where $\mathbf{0}$ is a $m \times n$ matrix of all zeros. For our module identification problem, we have the following proposition derived from (3.5):

Proposition 1. The subgradient of the objective function of our relaxed optimization problem $F(X)$ at the assignment X^t can be defined as: $\partial F(X^t) = 2(P-W)X^t\overline{Q}$. In our implementation, we choose

$$\overline{Q}_{rs} = \begin{cases} \alpha & Q_{rs} = 0; \\ 1 & Q_{rs} > 0; \\ -1 & Q_{rs} < 0, \end{cases} \tag{3.6}$$

where α is a number between $[-1, 1]$.

Proof: From (3.5), there always exists a $\overline{Q}$ satisfying $\left\|\overline{Q}\right\|_{L_\infty} \le 1$ and $\|Q\|_{L_1} =< \overline{Q}, Q >$. As $\partial \|Q\|_{L1} = \partial < \overline{Q}, Q >$ and the subgradient of differentiable functions is equal to its gradient [8], we have $\partial F(X^t) = -\partial \left[\|Q\|_{L1}\right] = -\partial < \overline{Q}, Q >= -\partial \mathrm{tr}(\overline{Q}^T X^{tT}(W-P)X^t) = 2(P-W)X^t\overline{Q}$ when X^t is close to local minima. QED.

Convex programming method

Using Frank-Wolfe algorithm with the derived subgradient, we now have a conditional subgradient method [8] to iteratively solve the relaxed optimization problem as shown in the pseudo-code given in the following Algorithm.

```
Algorithm: Conditional Subgradient
Input: initial value $X^t$, $t = 0$.
Do:
    (i)   Compute the subgradient $\partial F(X^t)$.
    (ii)  Solve the minimization problem:
              $X^* = \arg\min_X : \quad < \partial F(X^t), X > \quad s.t. \quad X \in \gamma$
    (iii) Linear search for the step in the direction $X^* - X^t$ found in (ii), update $X^t$, $t = t+1$.
Until: $|\triangle F| + \|\triangle X^t\| < \xi$
Output: $X^t$.
```

In this algorithm, step (ii) at each iteration can be solved using a generic linear programming solver in $O((qN)^{3.5})$. However, due to the special structure of the optimization problem, we instead solve it as a semi-linear assignment problem(because assignment matrix $[\partial F(X^t)]_{n \times q}$ is not a square matrix), which can be efficiently solved by assigning node i to module r, which is the index of the largest entry in row i of subgrident $\partial F(X^t)$ with the time complexity $O\left(n \times q\right)$.

To get the solution to the original problem (3.3) from the results of the relaxed problem by the conditional subgradient algorithm, we recover from the relaxed solution to a closest feasible solution by a simple rounding up strategy. Finally, we note that the presented conditional subgradient algorithm converges to a stationary point of the combinatorial optimization problem (3.3) [11] due to the non-convex nature of the objective function (3.2) with the worst case complexity $O\left(n^3\right)$. Hence, good initialization is critical to get high quality results. In our current implemention, we initialize X^t by a modified Expectation-Maximization (EM) algorithm presented in [71].

3.1.1.4 Path generation method(PG)

In order to make use of the local optima found by the fast subgradient method, we proposed a novel path generation method for our combinatorial optimization problem. The path generation method aims to conserve the overlaps between two local optima results, and get improvement based on the overlaps which make great contribution to the objective value. Our path generation is inspired by path relinking method which connects two combinatorial local optima and try to find better result along the connection [61]. However, our method is not to relink two local results but to create new paths by extracting useful overlaps between them.

The essential idea of the path generation method is to construct new results by preserving useful overlaps between modules from two local optima. Given two solutions x_A and x_B as the new path generators, PG generates new results and explores the search space on basis of maintaining the current productive overlaps between x_A and x_B. Let $N_r(x_A)$ denote the module u_r of x_A and $N_s(x_B)$ the module u_s of x_B. The contribution $X(r,s)$ by maintaing the overlap $Over(r_A, s_B)$ between $N_r(x_A)$ and $N_s(x_B)$ is defined as:

$$X(r,s) = \|s_{AB}^T(W-P)X_A\|_{L1} + \|s_{AB}^T(W-P)X_B\|_{L1} \tag{3.7}$$

in which s_{AB} is a binary vector, of which each element is equal to 1 when the corresponding node is in both $N_r(x_A)$ and $N_s(x_B)$, and 0 otherwise. The value of $X(r,s)$ is the shared contribution to the objective function Q^* in (3.1) between $N_r(x_A)$ and $N_s(x_B)$ in two feasible solutions. X_A and X_B are assignment matrix of the two solutions. Then the most promising overlap between modules r_A and s_B are determined by

$$(r_A, s_B) = argmax\{X(r,s) : r,s \in \{1,...,q\}\}. \tag{3.8}$$

The path generation based on (3.8) proceeds in the following manner: First, the most promising overlap $Over(r_A, s_B)$ between modules r_A and s_B of the initiating solution x_A and the guiding solution x_B are identified by (3.8), then r_A is locally adjusted to become $Over(r_A, s_B)$ by removing nodes. After the adjustment, a new solution x_1 is generated and $C_A = \{r_A\}$ and $C_B = \{s_B\}$, where C_A and C_B denote the sets of used modules in both solutions. Local search is then applied to find the improved x_1^*. Then we preserve x_1^* and let $x_A = x_1^*$ and repeats the above procedure until no overlap exists or other termination condition (for example, $N_{stop} = 5$ means that there are no larger than 5 nodes overlap modules exist in both solutions). In the end, we obtain the best solution along the generated results. The whole procedure is illustrated as following:

```
Algorithm: Path Generation (PG) Method
Input: $x_A$, $x_B$, $x$, $x_{best}$, $N_{stop}$, $C_A = \emptyset$, $C_B = \emptyset$, $Over = +\infty$, $Q_{best} = -\infty$
While($Over > N_{stop}$)
(1)   $(r_A, s_B) = argmax\{S(r,s) : r, s \in \{1, ..., q\}\}$ and find $Over(r_A, s_B)$;
(2)   modify nodes from $r_A$ in $x$ to make $N_r(x) = Over(r_A, s_B)$ and $C_A = \{r_A\}$, $C_B = \{s_B\}$;
(3)   $(Q_x^*, x^*) =$ LocalSearch($x$);
(4)   If ($Q_x^* > Q_{best}$)
(5)       $Q_{best} = Q_x^*$ and $x_{best} = x^*$;
(6)   EndIf
(7)   $x_A = x^*$ and find the next $Over$ set using (3.8);
EndWhile
Output: $x_{best}$ and $Q_{best}$.
```

To illustrate how PG works, an example of the path generation procedure is shown in Fig. 3.1. The modules organization of the given network is shown in Fig. 3.1A. Assume $x_A = \{\{1, 2, 4\}, \{5, 6\}, \{3, 7\}\}$ with $Q^*_{x_A} = 0.201$ and $x_B = \{\{1, 2\}, \{3, 4\}, \{5, 6, 7\}\}$ with $Q^*_{x_B} = 0.238$. Starting with $C_A = C_B = \emptyset$, a path is generated. At the first step, the most productive overlap between module $r_A = u_1^A$ in

x_A and $s_B = u_1^B$ in x_B is identified, and a new solution x_1 is obtained by modifying $r_A = u_1^A$ to be the same as $Over(u_1^A, u_2^B)$ with $Q_1 = 0.201$. Update $C_A = \{u_1^A\}$ and $C_B = \{u_1^B\}$. Local search further improves the solution to obtain x_1^* with $Q_1^* = 0.374$ and let then $x_A = x_1^*$. Next, module $r_A = u_2^A$ and $s_B = u_2^B$ have the largest contribution overlap. By similarly modifying $N_2(x_A) = Over(u_2^A, u_2^B)$, we then generate a path x_2^* with $Q_2^* = 0.374$. In the end, we make $N_3(x_A) = Over(u_3^A, u_3^B)$ and get the finally path x_3^* with $Q_3^* = 0.374$. The PG algorithm is executed with the time complexity $O(n^3)$.

3.1.2 Experimental results

We have implemented our SGPG method to identify functional modules in two biological networks: *Sce* PPI network from DIP [90] and *Hsa* PPI network from HPRD [82]. We first show the efficiency of SGPG comparing to these of SA for functional module identification in two networks with q = 10, 50 and 100. We further evaluate the potential of SGPG to identify biologically meaningful modules by contrasting the differences of the fine-grained modules ($q = 500$ for *Hsa* network and $q = 300$ for *Sce* network) detected by MCL algorithm [26], we show that SGPG can unearth certain kinds of biologically meaningful modules that may not be detected by MCL.

3.1.2.1 Comparison between SA and SGPG for coarse-grained modules

We first compare SA and SGPG for the analysis of two PPI networks for small number of modules ($q = 10, 50, 100$) due to the large computation time of SA when $q > 100$. The *Hsa* PPI network has a largest component of 9,270 nodes and 36,917 edges. The upper bound of the objective function value in (3.1) $Q^*_{max} = 0.98$ when we consider the original network itself as the image graph with $q =$ 9,270. We also have implemented our algorithm to the *Sce* PPI network, which has a largest component of 4990 nodes and 21,911 edges with the upper bound $Q^*_{max} = 0.97$ when $q = 4990$.

The parameters used by SA and SGPG are listed in Table 3.1. The starting temperature has been set sufficiently high for SA algorithms. The cooling down procedure in SA is slow enough to avoid freezing in metastable states (local optima). For SGPG, we set the local results number N_{set} to 10 (the number of local optima) and the terminal condition $N_{stop} = 5$ (when largest overlap set *Over* less than 5, then stop) and use path generation method to improve the solution quality.

Table 3.1: Parameter settings in SA and SGPG

Para.	C_β	T_{start}	T_{end}	T_{sweep}	T_{switch}	N_{set}	N_{stop}.
SA	0.99	40	0.001	100	20	-	-
SGPG	-	-	-	-	-	10	5

Table 3.2 gives the comparison of the fitting score Q^* given in (3.1) and the running time between SA and SGPG. The fitting score Q^* obtained by two algorithms with different q serve as the criterion of the solution quality or accuracy that reflects the mathematical optimality by (3.1) as we do not have the ground truth of actual functional modules in two networks. All the experiments are based on a C++ implementation on a MacPro Station with a 2.4GHz CPU and 6Gb RAM. From Table 3.2, the quality of the final solutions by SGPG is competitive to the results of SA with the largest gap 3.9% in Q^* with $q = 100$ for *Hsa*. At the same time, SGPG is consistently faster than SA. As shown in Table 3.2, computation time for both SA and SGPG grows quadratically with q but SGPG is significantly faster than SA especially when q become large. This makes a big difference when we need to identify a large number of modules with q increases over 200. For example, to identify $q = 300$ modules in the *Hsa* network, SA needs more than two months to finish one round computation

using the same settings in Table 3.1, while SGPG only needs around three days to obtain the results.

Table 3.2: Comparison of SA and SGPG on *Hsa* and *Sce*

PPI	Method	$Q^*(q{=}10)$	Time(h)	$Q^*(q{=}50)$	Time(h)	$Q^*(q{=}100)$	Time(h)
Hsa	SA	0.5393	1.73	0.6530	45.07	0.7180	180.26
	SGPR	0.5346	0.5	0.6452	1.95	0.6898	6.35
Sce	SA	0.5692	1.35	0.6834	25.02	0.7544	102.65
	SGPR	0.5690	0.3	0.6752	1.15	0.7292	3.34

To further investigate whether these two different block modeling based methods have the potential of identifying biologically meaningful modules, we perform Gene Onotolgy (GO) enrichment analysis for the identified modules using GoTermFinder [15]. Fig. 3.2 shows the comparison of the number of significantly enriched modules with different q using SA and SGPG. From both figures, SA has identified many GO enriched modules for all cases and SGPG achieves competitive performances.

3.1.2.2 Comparison between SGPG and MCL

In order to demonstrate the biological significance of module identification by block modeling, we have implemented both SGPG and MCL to detect fine-grained modules for *Hsa* and *Sce* networks. As SA will take months to obtain results with $q >= 200$, we only have implemented SGPG in this set of experiments. By analyzing the identified modules using two methods, we have found that SGPG not only can identify a competitive number of GO enriched modules as MCL does; but also can discover a number of biologically meaningful modules that MCL may fail to detect.

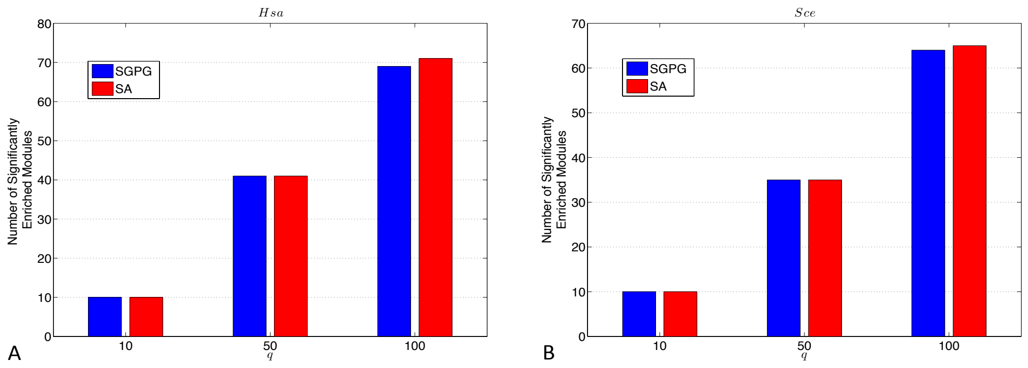


Figure 3.2: Comparison between SA and SGPG for the number of identified modules of *Hsa* PPI network(A) and *Sce* PPI network(B) that have significantly enriched GO terms below 1%.

3.1.2.3 Sce *PPI network*

We have identified fine-grained modules for the *Sce* network using SGPG and MCL. We set $q = 300$ for SGPG and the inflation parameter $I = 1.5$ for MCL, which identified 370 modules in total. Within 296 modules by SGPG and 307 modules by MCL that have more than two nodes, we have found 150 and 153 modules respectively with significantly enriched GO terms below 1% after Bonferroni-correction by GoTermFinder. SGPG performs competitive to MCL. But more important, we find that SGPG could detect sparsely connected modules with certain interaction patterns that MCL fails to detect.

In order to investigate the difference between the modules detected by SGPG and MCL. We have annotated them using KOG categories [104]. For each module, the KOG category is determined as the category assigned to the most proteins in it. Fig. 3.3A shows the percentage of the modules annotated to different KOG categories. The number of modules annotated to KOG categories U, K, J and T are clearly different (difference is larger than 2.5%) for both methods. Specifically, SGPG discovers more modules annotated to KOG U, K and T. To further investigate the cellular functionalities of different KOG categories, we find that proteins in

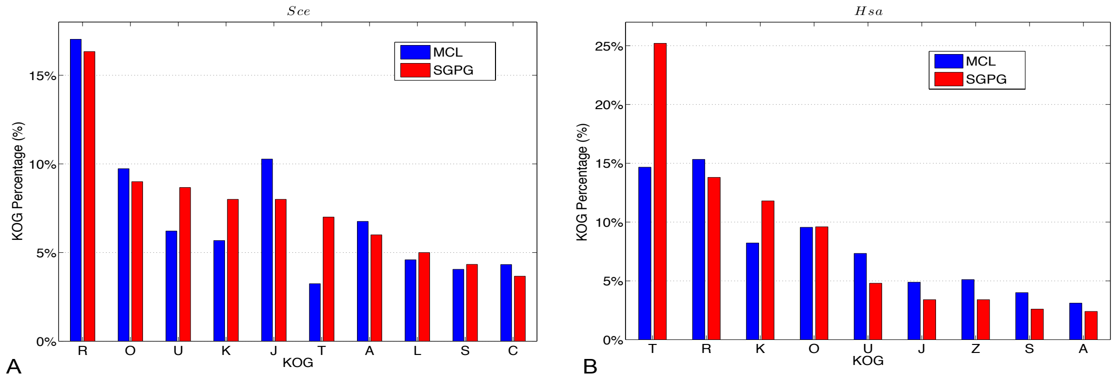


Figure 3.3: Percentage of different categories of modules by SGPG and MCL (annotated by KOG). A. KOG percentage of *Sce*. B. KOG percentage of *Hsa*.

KOG U play roles in intracellular trafficking, secretion, and vesicular transport, proteins in KOG K have functionalities in transcription and proteins in KOG T behave signal transduction. Proteins in KOG T and K have been shown to have sparsely connected functional modules structures [80]. Block modeling based SGPG has successfully detected more such modules with nodes sparsely connected but sharing similar interaction patterns comparing to MCL. For proteins in KOG J (functions in translation, ribosomal structure and biogenesis), they are supposed to have more cohesive modular structures with highly self-connected modules, which MCL tends to detect.

Table 3.3: Topological analysis of different KOG categories in *Sce* network

KOG ID	Method	proteins	sparse modules/modules	Avg. density	Avg. clustering coef.
U	SGPG	353	15/26	2.98%	0.0814
	MCL	256	0/21	27.38%	0.2402
K	SGPG	359	6/24	6.68%	0.1352
	MCL	361	0/19	26.35%0	0.1834
J	SGPG	579	9/24	9.16%	0.0678
	MCL	358	0/25	37.90%	0.1429
T	SGPG	169	13/21	3.47%	0.0755
	MCL	94	0/12	31.31%	0.0912

To validate that SGPG does discover sparsely connected functional modules, we examine the network topology of proteins in the modules annotated to KOG U, K, J and T. We have studied all the identified modules in these categories to count the number of sparsely connected modules, which the connection density within the module less than 3%. The detailed comparison is in Table 3.5. Table 3.5 lists the difference of network topology quantified by the average module density and the average clustering coefficient among proteins detected by SGPG and MCL. The average clustering coefficients of the subnetworks, which are induced from the original *Sce* network based on proteins of certain KOG categories in identified modules, are computed by the definition in [56]. Larger average clustering coefficients indicate that modules have modular structures with densely connected nodes [56]. From the table, the average clustering coefficients for modules detected by MCL are larger than those identified by SGPG. There is a similar trend for the average module density and modules discovered by MCL are more densely connected.

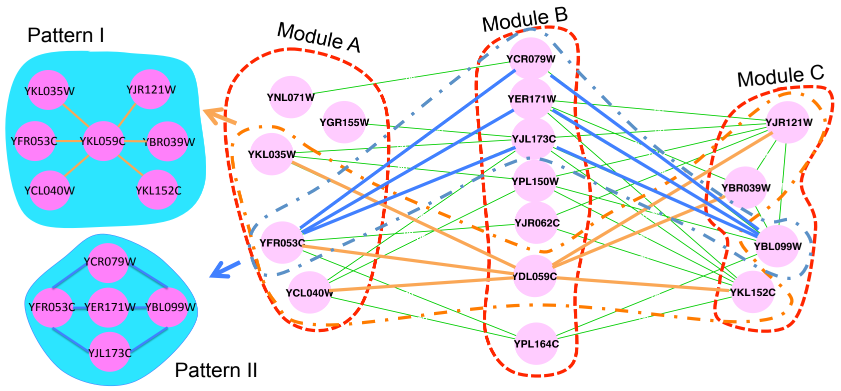


Figure 3.4: A subnetwork with sparsely connected modules detected by SGPG. Module A is enriched in hexokinase activity. Module B is enriched in response to endogenous stimulus. Module C is enriched in nucleoside phosphate metabolism.

Fig. 3.4 illustrates an induced subnetwork of sparsely connected modules discovered by SGPG from the *Sce* network. Only the interactions among the proteins in Fig. 3.4 are exhibited. As shown in Fig. 3.4, Module A, B are both sparsely connected modules, which have no interactions inside the modules. Module C is a cohesively connected module. Modules A, B and C are all significantly enriched in GO terms related to KOG G(carbohydrate transport and metabolism), T (signal transduction mechanisms) and C(energy production and conversion) respectively. From the structure of the Fig. 3.4, we find that proteins in module B play the role in passing signal between proteins of hexokinase activity and nucleoside phosphate metabolism. Furthermore, we notice that marked patterns I and II are two types of interaction patterns across these modules, which tend to be grouped into the same module when using MCL. Fig. 3.4 clearly displays the advantage of SGPG, which is to identify modules by their interaction patterns and functional roles rather than their interaction density.

In addition, Table 3.4 lists three module examples, including module B in Fig. 3.4, detected by SGPG but missed by MCL. These three modules are annotated to KOG U and T respectively. The common property of these three modules is that they are all sparsely connected, which is the reason that MCL fails to detect this type of modules as MCL tends to identify highly self-connected modules [51]. In order to thoroughly check whether MCL is capable of detecting these three modules, we have tuned the inflation parameter I from 1.4 to 5.0 to run MCL several times. However, no matter which inflation parameter we choose, MCL cannot detect these three sparsely connected modules.

3.1.2.4 Hsa *PPI network*

For the *Hsa* network, we set SGPG to identify $q = 500$ modules with the same settings in Table 3.1. For MCL, we set its inflation parameter $I = 1.5$ and have found 450 modules. Because most of these identified modules have more than two nodes

Table 3.4: Sparse modules in O, U and T KOG categories for *Sce* network

KOG ID	Sparse module example	Enriched GO Term	GO Level	p-value
U	YDR179C, YNL287W, YDL216C YCR099C, YIL004C,YAL026C YLR268W, YLR093C, YPR163C YPR148C, YOL064C, YOL117W YGL084C, YLR031W, YIL076W YPL179W, YKL191W, YPL010W	protein deneddylation	[+8, 0]	2.01e-5
T	YJL092W, YDR490C, YOR231W YJL005W, YPL074W, YPL083C YNL323W,YOL100W	signal transduction	[+3, -1]	6.09e-5
T	YDR076W, YDL059C, YJL173C YPL164C, YER171W, YPL026C YCR079W, YPL150W, YHR169W YJR062C	response to endogenous stimulus	[+2, -1]	4.77e-5

(478 from SGPG and 380 from MCL), we have performed GO enrichment analysis for only modules with more than two nodes by both SGPG and MCL. By GoTermFinder, 269 modules from SGPG and 265 modules from MCL are significantly enriched below 1% after Bonferroni-correction. SGPG has discovered a competitive number of GO enriched modules compared to MCL. We also note that the modules identified by SGPG are relatively smaller than those from MCL and these modules have more specific enriched functionalities and may provide more detailed information for future catalog of functional modules. More importantly, SGPG detects several modules with interesting functionalities that MCL has missed.

Following the same analysis method used in section (3.2.1), we first annotated all identified modules by KOG category to scrutinize the difference between modules detected by SGPG and MCL. Fig. 3.3B shows the percentage of the modules annotated to different KOG categories by both methods. Obviously, SGPG detects more modules annotated in KOG T and K, within which functional modules tend to have sparsely connected structure. However, MCL discovers more modules annotated in KOG U, within which functional modules tend to have densely connected structure.

Table. 3.5 further consolidates that the modules detected by SGPG have more sparsely connected patterns than MCL. The average density and average clustering coefficient both indicate that modules discovered by MCL have cohesive modular

Table 3.5: Topological analysis of different KOG categories in *Hsa* network

KOG ID	Method	proteins	sparse modules/modules	Avg. density	Avg. clustering coef.
T	SGPG	1970	59/126	4.91%	0.0822
	MCL	2481	0/66	26.32%	0.1696
K	SGPG	878	27/59	3.15%	0.0779
	MCL	916	0/37	30.41%0	0.1928
U	SGPG	592	3/24	4.95%	0.0448
	MCL	517	0/33	31.42%	0.1359

structure, while modules discovered by SGPG are more sparsely connected.

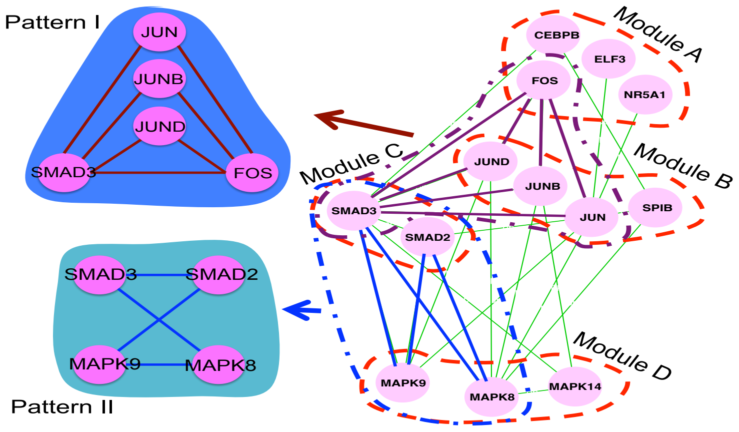


Figure 3.5: A subnetwork with sparsely connected modules detected by SGPG. Module A is enriched in sequence-specific DNA binding with. Module B is enriched in cellular response to calcium ion. Module D is enriched in MAP kinase activity.

Fig. 3.5 illustrates an induced subnetwork discovered by SGPG from the *Hsa* network. Only the interactions among the proteins in Fig. 3.5 are exhibited. As shown in Fig. 3.5, Module A, B and C are all sparsely connected modules, which have no interaction inside the modules. Proteins in module D only have a few connections. Module A and B are annotated to KOG K(transcription). While module D is annotated to KOG T(signal transduction mechanisms). Module C is annotated to both KOG T and K. Module C contains proteins SMAD2 and SMAD3 which play the important role in tumor formulation [57]. From the module structure in Fig. 3.5, we find that SMAD2 and SMAD3 have intimate relationship to proteins of DNA binding, cellular response and kinase activity, which is useful to help us to have a better understanding of their functionality and influence on other proteins.

Table 3.6: Sparse modules in O, U and T KOG categories for *Sce* network

KOG ID	Sparse module example	Enriched GO Term	GO Level	*p*-value
T	NTRK1, NTRK3, NTRK2 VAV1, VAV3	neurotrophin receptor activity	[+3, -1]	2.95e-9
T	PIK3R3, PIK3R2, PIK3R1	phosphatidylinositol 3-kinase complex	[+5, -1]	4.77e-9
K	JUN, JUNB, JUND SPIB	cellular response to calcium ion	[+6, -1]	4.04e-7

Table 3.6 lists three sparsely connected module examples detected by SGPG but missed by MCL. These three modules are annotated to KOG T and K respectively, which cannot be detected by MCL no matter what inflation parameter we choose.

3.1.3 Conclusions

We have proposed a novel efficient method SGPG that combines SG and PG to solve block modeling module identification problem. Our experimental results have demonstrated that block modeling based methods are superior to other state-of-the-art algorithms. Furthermore, our SGPG method can achieve competitive clustering performance as the original SA method efficiently. We also have shown that SGPG can detect functional modules with biological significance, especially sparsely connected modules, which carry important cellular functionalities.

3.2 Two hop random walk

In this section, we propose a novel formulation to solve the functional module identification problem, which simultaneously identifies the previously described dense and sparse modules with similar interaction patterns. The section is organized as follows: In section 3.2.1, we first introduce the new optimization formulation by searching for the low two-hop conductance sets (LCP^2) based on the two-hop transition matrix of the underlying Markov chain of the random walk on a given network. Then, we derive the corresponding mathematical programming problem and propose an algorithm SLCP^2, which solves LCP^2 to search for non-overlapping modules by a spectral approximate method with a close-to-optimal solution. We also present an extended algorithm GLCP^2, which solves LCP^2 to search for overlapping modules by a bottom-up greedy strategy. In section 3.2.2, we evaluate and compare our methods with other state-of-the-art algorithms for functional module identification on four large-scale PPI networks: the *Saccharomyces cerevisia* PPI network extracted from the Database of Interacting Proteins (DIP) [90] (*Sce*DIP); the corresponding network from the BioGRID database [101, 17] (*Sce*BioGRID); the *Homo sapiens* (*Hsa*HPRD) PPI network collected from the Human Protein Reference Database (HPRD version 9) [82]; and the human PPI network *Hsa*BioGRID obtained from BioGRID [101, 17]. The experimental results of protein complex prediction show that non-overlapping SLCP^2 outperforms most of the non-overlapping state-of-the-art algorithms and performs competitively with the more recent RMCL algorithm [91, 93]. When we compare GLCP^2 with the other algorithms for overlapping modules, our experiments show that GLCP^2 outperforms ClusterOne [68] and LinkComm [1]. High level GO (Gene Ontology) term [7] prediction results further demonstrate that SLCP^2 is superior to other non-overlapping algorithms while GLCP^2 and LinkComm perform equally well. Furthermore, we present a few identified functional sparse modules to illustrate that SLCP^2 and GLCP^2 have the advantage in detecting functional sparse modules compared to the other state-of-the-art algorithms in the last part of section 3.2.2. In section 3.2.3, we draw our conclusions by briefly summarizing the differences between our new SLCP^2 and GLCP^2 algorithms and other existing module identification algorithms.

3.2.1 Methodology

For random walk on G, its underlying Markov chain can be characterized by a transition matrix $P = D^{-1}A$, where $D = Diag(d_1, d_2, ..., d_n)$ is an $n \times n$ diagonal matrix with the corresponding node degrees ($d_i = \sum_j A_{ij}, i = 1, ..., n$) on its diagonal. As G is connected, the underlying Markov chain of the random walk is irreducible and ergodic and therefore there exists a stationary distribution satisfying $P^T\pi = \pi$, where $\pi_i = d_i/M, M = \sum_{i=1}^{n} d_i$. The conductance of a subset of nodes C in G has been defined as [43]

$$\Phi_P(C, \bar{C}) = \frac{\sum_{i \in C, j \in \bar{C}} \pi_i P_{ij}}{\sum_{i \in C} \pi_i}, C \cup \bar{C} = V, \tag{3.9}$$

Finding k low conductance (LC) sets in the network G based on this conductance definition involves partitioning the node set V into k subsets $(C_1, C_2, ..., C_k)$, which can be formulated as the following optimization problem:

$$\min \sum_{h=1}^{k} \Phi_P(C_h, \bar{C_h}) \quad s.t. \bigcup_{h=1}^{k} C_h = V; C_h \cap C_l = \oslash, h \neq l. \tag{3.10}$$

We call this method LCP for simplicity and LCP is equivalent to the formulation of normalized k-cut in [115].

3.2.1.1 Interaction patterns and transition matrix P^2

Considering Markov random walk on the given network G, its corresponding transition matrix P describes the transition probability that the random walker walks from one node to another in one step. With two directly interacting nodes ($A_{ij} = 1$), the corresponding transition probability is uniformly random among all the direct neighbors: $P_{ij} = \frac{A_{ij}}{d_i}$, denoting the probability of walking from node i to j

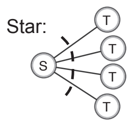
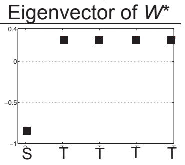
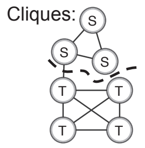
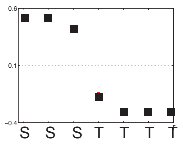
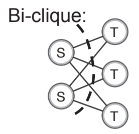
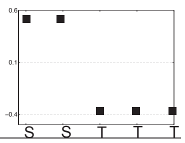
Motifs	P	LCP	P^2	LCP^2	2nd Largest Eigenvector of W^*
	$\begin{array}{ccccc:c} & S & T & T & T & T \\ S & 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ T & 1 & 0 & 0 & 0 & 0 \\ T & 1 & 0 & 0 & 0 & 0 \\ T & 1 & 0 & 0 & 0 & 0 \\ \hdashline T & 1 & 0 & 0 & 0 & 0 \end{array}$	1.14	$\begin{array}{cc:cccc} & S & T & T & T & T \\ S & 1 & 0 & 0 & 0 & 0 \\ \hdashline T & 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ T & 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ T & 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ T & 0 & 0.25 & 0.25 & 0.25 & 0.25 \end{array}$	0	
	$\begin{array}{cccc:cccc} & S & S & S & T & T & T & T \\ S & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ S & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ S & 0.33 & 0.33 & 0 & 0.33 & 0 & 0 & 0 \\ \hdashline T & 0 & 0 & 0.25 & 0 & 0.25 & 0.25 & 0.25 \\ T & 0 & 0 & 0 & 0.33 & 0 & 0.33 & 0.33 \\ T & 0 & 0 & 0 & 0.33 & 0.33 & 0 & 0.33 \\ T & 0 & 0 & 0 & 0.33 & 0.33 & 0.33 & 0 \end{array}$	0.22	$\begin{array}{cccc:cccc} & S & S & S & T & T & T & T \\ S & 0.42 & 0.17 & 0.25 & 0.17 & 0 & 0 & 0 \\ S & 0.17 & 0.42 & 0.25 & 0.17 & 0 & 0 & 0 \\ S & 0.17 & 0.17 & 0.42 & 0 & 0.08 & 0.08 & 0.08 \\ \hdashline T & 0.08 & 0.08 & 0 & 0.33 & 0.17 & 0.17 & 0.17 \\ T & 0 & 0 & 0.08 & 0.22 & 0.31 & 0.19 & 0.19 \\ T & 0 & 0 & 0.08 & 0.22 & 0.19 & 0.31 & 0.19 \\ T & 0 & 0 & 0.08 & 0.22 & 0.19 & 0.19 & 0.31 \end{array}$	0.31	
	$\begin{array}{cc:cccc} & S & S & T & T & T \\ S & 0 & 0 & 0.33 & 0.33 & 0.33 \\ \hdashline S & 0 & 0 & 0.33 & 0.33 & 0.33 \\ T & 0.5 & 0.5 & 0 & 0 & 0 \\ T & 0.5 & 0.5 & 0 & 0 & 0 \\ T & 0.5 & 0.5 & 0 & 0 & 0 \end{array}$	1.33	$\begin{array}{ccc:ccc} & S & S & T & T & T \\ S & 0.5 & 0.5 & 0 & 0 & 0 \\ S & 0.5 & 0.5 & 0 & 0 & 0 \\ \hdashline T & 0 & 0 & 0.33 & 0.33 & 0.33 \\ T & 0 & 0 & 0.33 & 0.33 & 0.33 \\ T & 0 & 0 & 0.33 & 0.33 & 0.33 \end{array}$	0	

Figure 3.6: Different module identification results obtained by using P and P^2. The 1st column displays three basic motifs (star motif, clique motif and bi-clique motif) (used by [86]) and the black dashed lines show the natural partitions. The 2nd column gives the P of three basic motifs and the black dashed lines denote the module dividing lines obtained by LCP. The 3rd column gives the minimum objective function values by (4.26). The 4th column gives the P^2 of three basic motifs and the black dashed lines indicate the identified modules by LCP^2. The 5th column shows the minimum objective function values based on (3.11). The last column illustrates the 2nd largest eigenvector of W^* used in Algorithm 1.

in one step. Clearly, nodes without connections have no chance to reach each other in one step. The conductance definition in (4.27) extends to the transition probabilities between two complement partitions C and $\bar{C}$ in the given network. Hence, finding low conductance sets defined by P (LCP) tends to find densely connected modules as it aims to minimize the transition probabilities between potential modules to the rest of the network, which are dependent on the corresponding cut size or the number of edges across potential modules.

However, in addition to densely connected modules, functional module identification in PPI networks desires to detect other meaningful modules with nodes having similar interaction patterns in networks. The star and bi-clique motifs in Fig. 3.6 show that nodes with similar interaction patterns may be sparsely connected or even have no interactions among them. For example, nodes marked by "S" and "T", which should be grouped into two respective modules, all have the same interaction patterns based on the network structure. But because there are no interactions among them, existing algorithms for densely connected modules, including LCP, rarely cluster them into the corresponding modules correctly. The second column in Fig. 3.6 lists the random walk transition matrix P of each motif and the module dividing lines by LCP derived based on P. The third column in Fig. 3.6 gives the objective function values computed by LCP (4.26). Based on the analysis of the three basic motifs, we confirm that LCP only focuses on detecting dense modules, which may not be adequate for functional module identification in PPI networks.

In order to identify modules of more diverse topology based on interaction patterns, we propose to search for low conductance sets defined by a two-hop transition matrix $P^2 = P \times P$ (LCP^2). Intuitively, nodes with similar interaction patterns (no matter whether densely connected or sparsely connected) are more likely to transit back to the nodes in the same module after two steps of random walk. Therefore, we redefine the conductance by replacing P with P^2, which captures more meaningful modular structures in PPI networks. The fourth and fifth columns in Fig. 3.6 show P^2 transition matrices and module dividing lines for three basic motifs and low conductance values computed by P^2, respectively. From P^2 in Fig. 3.6, we find that the nodes with the same interaction patterns have higher probabilities to walk to each other in two random walk steps. Therefore, the correct module identification of star and bi-clique motifs can be achieved by finding low conductance sets defined by the two-hop transition matrix P^2. For the clique motif, the nodes in cliques still have the same interaction patterns though the low conductance value computed by P^2 increases. Therefore, the corresponding cliques can still be correctly identified by LCP^2 as potential modules. The example of these three motifs demonstrates that dense modules like cliques and sparse modules such as stars and bi-cliques can be identified simultaneously through searching for low conductance sets based on P^2.

Based on these motivating examples, finding low conductance sets using P^2 has the promising potential to discover biologically meaningful modules consisting of the nodes with similar interaction patterns. We now provide the mathematical formulation and the optimization algorithm to solve LCP^2.

Similar to LCP, we aim to solve the following minimization problem LCP^2 by using the two-hop transition matrix P^2:

$$\min \sum_{h=1}^{k} \Phi_{P^2}(C_h, \bar{C_h}) \quad s.t. \bigcup_{h=1}^{k} C_h = V; C_h \cap C_l = \oslash, h \neq l. \tag{3.11}$$

in which $\Phi_{P^2}(C_h, \bar{C_h})$ is the new conductance based on P^2. Note that P^2 is still a stochastic matrix and its stationary distribution is also π ($P^T P^T \pi = P^T \pi = \pi$). We can derive that $\Phi_{P^2}(C, C) + \Phi_{P^2}\left(C, \bar{C}\right) = 1$. With these, the above problem (3.11) can be transformed to an equivalent formulation:

$$\max \sum_{h=1}^{k} \Phi_{P^2}(C_h, C_h) \quad s.t. \bigcup_{h=1}^{k} C_h = V; C_h \cap C_l = \oslash, h \neq l. \tag{3.12}$$

As the underlying Markov chain is ergodic given a connected network, we have $\pi_i P_{ij} = \pi_j P_{ji} = A_{i,j}/M$ and $\pi_i = d_i/M$. By expanding the objective function in (3.12), we can further derive

$$\begin{aligned}
&\sum_{h=1}^{k} \Phi_{P^2}\left(C_h, C_h\right) \\
&= \sum_{h=1}^{k} \frac{\sum_{i,j \in C_h} \pi_i P_{ij}^2}{\sum_{i \in C_h} \pi_i} = \sum_{i=1}^{k} \frac{\sum_{i,j \in C_h} \pi_i \sum_{l=1}^{n} P_{il} P_{lj}}{\sum_{i \in C_h} \pi_i} \\
&= \sum_{h=1}^{k} \frac{\sum_{i,j \in C_h} \sum_{l=1}^{n} A_{il} P_{lj}}{\sum_{i \in C_h} d_i} = \sum_{h=1}^{k} \frac{x_h^T A P x_h}{x_h^T D x_h} \\
&= \sum_{h=1}^{k} \frac{x_h^T A D^{-1} A x_h}{x_h^T D x_h} = trace\left(\frac{X^T A D^{-1} A X}{X^T D X}\right).
\end{aligned}$$

where x_h denotes the hth column of the $n \times k$ module assignment matrix X, which lies in the space:

$$\mathcal{F}_k = \{X : X1_k = 1_n,\ x_{ij} \in \{0,1\}\}, \tag{3.13}$$

in which 1_k and 1_n are vectors with all of their elements equal to 1.

Combining the transformed objective function and the constraint set (3.13), we can express LCP^2 as the following optimization problem:

$$(F)\begin{cases} \text{max:} & trace\left(\frac{X^TAD^{-1}AX}{X^TDX}\right) \\ s.t. & X \in \mathcal{F}_k \end{cases} \tag{3.14}$$

3.2.1.2 Module identification by interaction patterns

Non-overlapping Algorithm

We can further transform the problem (F) to the following relaxed optimization problem:

$$(F1)\begin{cases} \max & trace\left(Y^TWY\right) \\ s.t. & Y^TY = I_k, \end{cases} \tag{3.15}$$

where $W = D^{-1/2}AD^{-1}AD^{-1/2}$; and $Y = D^{1/2}X\left(X^TDX\right)^{-1/2}$ denotes the relaxed assignment matrix, which is orthonormal. Let $H = D^{-1/2}AD^{-1/2}$. We can rewrite $W = HH^T$ as the inner-product of H. Taking each column of H as the normalized interaction pattern of the corresponding node, this Gram matrix W measures the interaction similarity among different nodes (we note that the inner-product can be replaced by a general Mercer kernel if needed). According to this inner-product form of W, nodes in dense modules have high similarities as they share the same interaction pattern, which is to interact with each other within modules. At the same time, similarities among nodes in sparse modules are high because they interact with similar neighbors in the rest of the network. Consequently, similarities among nodes with similar interaction patterns (no matter whether in dense or sparse modules) are higher. Therefore, nodes that play identical roles in the network can be grouped together.

We note that a formulation similar to ours has also been independently presented in [92]. The authors in [92] have proposed to use a symmetrization strategy AA^T to detect interaction patterns of nodes. In our new LCP^2 formulation, module identification depends on the different form HH^T, which can be viewed as the normalized version of AA^T. As shown in previous results obtained by normalized cuts, we expect that this new formulation depending on the normalized version HH^T may yield more balanced modules that may lead to biologically meaningful functional module identification results.

In order to derive the solution strategy for LCP^2, we relax Y to be an orthonormal matrix and it turns out that $(F1)$ has a closed-form solution based on Ky Fan Theorem. (Ky Fan Theorem) Let T be a symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ and the corresponding eigenvectors $U = [u_1, ..., u_n]$. Then $\sum_{i=1}^{k} \lambda_i = \max_{X^T X = I_k} trace(X^T T X)$. Moreover, the optimal X^* is given by $X^* = [u_1, ..., u_k]Q$ with Q being an arbitrary orthogonal matrix.

Following this theorem, we can use the largest k eigenvectors of the Gram matrix W to approximate the module assignment matrix Y. Therefore, we propose our module identification algorithm SLCP^2 in Algorithm 1.

The 1st step in the algorithm aims to compute the interaction similarity more accurately by considering the self connection. Adding self loop can make dense modules more distinguishable and avoid impairing the dense modular structure by considering interaction patterns. The 2nd step computes W. The 3rd step removes the diagonal part of the Gram matrix W in order to get rid of the influence of

```
Algorithm 1 (Non-overlapping): Spectral Algorithm for LCP$^2$ (SLCP$^2$)
Input: Adjacency matrix $A$ and the number of modules $k$
Output: Module assignment matrix $X_n$
1. Add self loop to $A = A + I_n$
2. Compute $W = D^{-1/2}AD^{-1}AD^{-1/2}$
3. $W^* = W - Diag(diag(W))$
4. Find the largest $k$ eigenvalues and their corresponding eigenvectors $[E, V_k] = eig(W^*, k)$
5. Obtain the approximated module assignment $R$ by pivoted QR decomposition: $V_k^T P = Q[R11,\ R12]$, then $R = [I_k\ R_{11}^{-1}R_{12}]P^T$
6. The module membership of each node is determined by the row index of the largest element in the absolute values of the corresponding column of $R$
```

self similarity because proteins tend to be clustered into single node modules when they have large self similarities. In order to obtain modules of appropriate size, removing self similarity is necessary. The 4th step obtains the k largest eigenvectors of W^*. Steps 5 and 6 use the pivoted QR decomposition to approximate the module assignment matrix X [118]. The pivoted QR decomposition is a better option than the classic k-means method. It is well known that the performance of k-means heavily depends on its initialization. However, when dealing with a large-scale network that may have thousands of potential modules, it is difficult for k-means to find good initializations. Using the pivoted QR decomposition avoids the initialization step, therefore better performance can be achieved. As illustrated, the last column of Fig. 3.6 exhibits the second largest eigenvector of W^*, from which we can easily distinguish the two different modules in the three motifs in Fig. 3.6.

Overlapping Algorithm

Based on the previously derived Gram matrix W which contains the information of interaction similarity among all the nodes in the given network, we can further derive a bottom-up greedy algorithm to identify overlapping functional modules. The procedure of the greedy algorithm is illustrated in Algorithm 2. The idea of adopting the greedy strategy is similar to the one used in ClusterOne [68] to grow each module from each single protein as a seed. For each iteration, we add proteins to modules to acquire the most gain in the weight density of a module h, which can be computed as

$$W_d\left(C_h\right) = \frac{\sum_{i,j \in C_h} W_{i,j}}{|C_h|^2} \tag{3.16}$$

where $W_{i,j}$ measures the interaction similarity between protein i and j. We keep adding proteins to potential modules until there is no increase of the weight density.

Algorithm 2 (Overlapping): Greedy algorithm for LCP^2 (GLCP^2)

Input: Gram matrix W

Output: Module assignment matrix X_o

```
1. Assign each protein in its own module
2. Compute the average weight density $Q = \frac{\sum W_d(C_h)}{n}$
3. while($Q > \xi$)
4.         Chuffle protein list $V$
5.         for $i = 1 : |V|$
6.                 Add the protein $V_i$ to existing module $h$ to achieve the largest
7.                 positive weight density gain.
8.         endfor
9.         Re-compute the average weight density $Q$.
10. endwhile
11. Post-processing the obtained modules.
```

The post-processing step in Line 11 of Algorithm 2 aims to remove low quality modules and merge highly overlapped modules. Because our LCP^2 formulation can detect both densely connected modules and sparsely connected modules (the sparsely connected modules contain proteins with similar interaction to the rest of the network), we use two quality functions to evaluate the obtained modules. One quality function is qf_d = edge density $\times$ sqrt(size), which has been similarly adopted in [96] to identify high quality dense modules. The other quality function is qf_s = #.shared proteins/size for sparse modules. We remove the modules when

$qf_d < \alpha$ and $qf_s < \beta$, where α and β are two user-specified thresholds. With larger α and β, we may remove a larger number of low quality modules by qf_d and qf_s. After removing low quality modules, we merge highly overlapped modules based on $NA(a,b) = \frac{|V_a \cap V_b|^2}{|V_a| \times |V_b|}$, where a, b are two modules. If $NA(S_i, S_j) > p$, we merge modules S_i and S_j together. Here, p is another tuning parameter and we typically set it over 0.9 to guarantee that only highly overlapped modules are merged.

3.2.2 Experimental results

We first introduce how we implement the algorithms that we take for performance comparison; where we obtain the PPI networks and protein complex golden standard sets; and what criteria we use to evaluate the performance of the selected algorithms. After that, we compare all algorithms on synthetic networks with both dense and sparse module structures and show that both the non-overlapping and overlapping algorithms (SLCP^2 and GLCP^2 respectively) based on the two-hop transition matrix outperform all other state-of-the-art methods. Then, we analyze the performance of protein complex and high level GO term predictions to demonstrate the potential of predicting biologically meaningful modules by all compared algorithms. In the end, we illustrate that the algorithms based on our LCP^2 formulation are superior to the state-of-the-art algorithms in identifying sparse functional modules by displaying the module detection results for several specific biological functional sparse modules.

3.2.2.1 Algorithms, data and metric

Algorithms

For algorithms that identify non-overlapping modules, we compare SLCP^2 with five state-of-the-art algorithms, which are LCP [115], MCL (Markov Clustering algorithm) [26], RMCL (regularized MCL) [91, 93], GS (Graph Summarization) [67] and PG (Power Graph) [86]. Comparing with LCP aims to show that finding low conductance sets through P^2 is superior to LCP based on the conductance definition by P as LCP only focuses on detecting dense modules. We also compare SLCP^2 with MCL and RMCL because they are widely used network clustering algorithms in biological network analysis and have been shown to give biologically meaningful results. Additionally, two other algorithms, GS and PG, are chosen as they search for modules based on interaction patterns and hence are also able to detect both dense and sparse modules as SLCP^2 does.

For overlapping module identification algorithms, we compare our GLCP^2 with two other recently proposed algorithms: ClusterOne [68] and LinkComm (Link Community) [1]. In order to distinguish non-overlapping and overlapping algorithms, we mark all the overlapping algorithms with a star (*) in all the figures in our experimental results.

As discussed earlier, LCP is equivalent to the normalized k-cut problem [115]. Therefore, we adopt the spectral method proposed in [115] to solve LCP. The implementation of the k-means clustering algorithm used by LCP is based on the procedure proposed in [12]. We have obtained the source code for MCL [†], RMCL [‡], GS [§], PG [¶], ClusterOne [‖], and LinkComm [**] from the Web pages provided in the corresponding papers.

For non-overlapping module identification algorithms, SLCP^2 and LCP have one parameter k (the number of modules) and MCL also has one tuning parameter called "Inflation" I_F. RMCL has two tuning parameters, which are "balance" b and "Inflation" I_F. For the number of modules k in SLCP^2 and LCP, we implement

[†] http://www.micans.org/mcl

[‡] http://www.cse.ohio-state.edu/ satuluri/research.html

[§] https://open-innovation.alcatel-lucent.com/projects/gscode/

[¶] http://www.biotec.tu-dresden.de/re-search/schroeder/powergraphs/

[‖] http://www.paccanarolab.org/cluster-one/index.html

[**] https://github.com/bagrow/linkcomm

the grid search from $k = 500$ to 3000 with an interval of 100. For I_F in MCL, we similarly search from 1.2 to 5.0 with an interval of 0.1. For RMCL, we set b and I_F to 0.5 and 2.0 respectively based on the suggestions in the papers [93, 96]. Because both PG and GS are hierarchical bottom-up algorithms, they do not have any tuning parameter.

For overlapping module identification algorithms, LinkComm has one parameter t and GLCP^2 has three parameters α, β and p. For LinkComm, we set the threshold $t = 0.2$ as it yields the best results in our experiments. For GLCP^2, the parameters set (α, β, p) determines the quality of the results. From our experience, $(\beta, p) = (0.8, 0.9)$ gives good performance. As to α, it depends on the density of the original network. In the following experiments, we set $\alpha = 0.76$ for the *Sce* PPI networks and $\alpha = 0.7$ for the *Hsa* PPI networks, because the *Hsa* PPI networks are more sparse than the *Sce* PPI networks.

Table 3.7: Information of the four real-world PPI networks.

Network	#. nodes	#. edges	MIPS	SGD	PCDq	CORUM	$\|GO\|$
*Sce*DIP	4980	22076	203	305	—	—	1166
*Sce*BioGRID	5640	59748	203	305	—	—	1172
*Hsa*HPRD	9269	36917	—	—	1204	1294	4452
*Hsa*BioGRID	14283	87397	—	—	1204	1294	4457

The networks are the largest components of the original datasets. $|GO|$ is the number of GO terms whose information content is larger than 2.

Data

We have run all these selected algorithms on four PPI networks. Two of them are *Saccharomyces cerevisia* (*Sce*) PPI networks obtained from the DIP (Database of Interacting Proteins) [90] (SceDIP) and BioGRID database [17, 101] (SceBioGRID), respectively. The other two are the *Homo sapiens* (*Hsa*) PPI networks extracted from HPRD (Human Protein Reference Database) [82] (HsaHPRD) and BioGRID database [17, 101] (HsaBioGRID), respectively. We use the largest components of these four networks as the input of the algorithms.

We evaluate the complex prediction performance of the algorithms based on four protein complex golden standards. For *Sce* PPI networks, we use MIPS [63] and SGD [37] golden standards. For *Hsa* PPI networks, we adopt the PCDq [42] as well as CORUM (Comprehensive Resource of Mammalian protein complexes) golden standards [87] for our performance evaluation. We use all golden standard protein complexes with two or more proteins in all our experiments.

For examining whether the detected modules capture protein functional relationships other than just protein complexes, we use the high-level GO terms in all three domains (molecular function (F), biological process (P) and cellular component (C)) as the golden standard for GO term prediction. Any GO term, whose information content (IC) [96] is higher than two, is considered as a high-level GO term. The definition of the information content of a GO term g is IC $= -\log(|g|/|root|)$ as given in [96], where "*root*" is the corresponding root GO term (either F, P or C) of g. In addition, we remove GO terms which contain fewer than two proteins. The detailed information of the networks, complex golden standards and GO terms are listed in Table 3.7.

Metric

To evaluate the performance for complex prediction, we use two independent quality measures (used by [69]) to assess the similarity between the predicted complexes and the golden standard reference complexes. In our experiments, we set the minimum size of detected modules to three for fair comparison between all competing algorithms. The first measure counts the number of predicted modules matched to the golden standard reference modules. A predicted module a with V_a proteins is considered a match to a reference module b with V_b proteins when the neighborhood affinity $NA(a,b) = \frac{|V_a \cap V_b|^2}{|V_a| \times |V_b|} \geq 0.25$ [53, 69]. The threshold of 0.25 is chosen because it represents the case when at least half of the complexes overlap if the two compared complexes are equally large. The second measure is the geometric mean of two other measures, which are the cluster-wise sensitivity (Sn) and the cluster-wise positive predictive value (PPV) [53]. Given r predicted and s reference complexes, let t_{ij} denote the number of proteins that exist in both predicted complex i and reference complex j, and w_j represent the number of proteins in reference complex j. Then Sn and PPV can be defined as

$$Sn = \frac{\sum_{j=1}^{s} \underset{i=1,\ldots,r}{max}\, t_{ij}}{\sum_{j=1}^{s} w_j} \ , \ PPV = \frac{\sum_{i=1}^{r} \underset{j=1,\ldots,s}{max}\, t_{ij}}{\sum_{i=1}^{r}\sum_{j=1}^{s} t_{ij}}$$

Since Sn can reach its maximum by grouping all proteins in one module, while PPV can be maximized by putting each protein in its own module, we use their geometric mean as "accuracy" to balance these two measures ($Acc = \sqrt{Sn \times PPV}$) [69, 53].

To investigate the functional significance of identified modules, we follow the same strategy in [96] to compute F measure based on high-level GO term prediction.

Let $C = \{c_1, c_2, ..., c_k\}$ denote the identified modules and $G = \{g_1, g_2, ..., g_l\}$ denote the selected GO terms. We can calculate the number of identified modules that match at least one GO term, denoted by N_{cp}: $N_{cp} = |\,\{c_i \in C | NA(c_i, g_j) > 0.25, \exists g_j \in G\}\,|$. The number of GO terms that match at least one identified module can be computed: $N_{cg} = |\,\{g_i \in G | NA(c_i, g_j) > 0.25, \exists c_i \in C\}\,|$. Based on these numbers, we can further compute precision and recall: precision $= \frac{N_{cp}}{|C|}$, recall $= \frac{N_{cg}}{|G|}$. The final F-measure is the harmonic mean of precision and recall: $F = 2 \times \text{precision} \times \text{recall}/(\text{precision} + \text{recall})$.

Finally, all experiments illustrated in this section can be accomplished within one

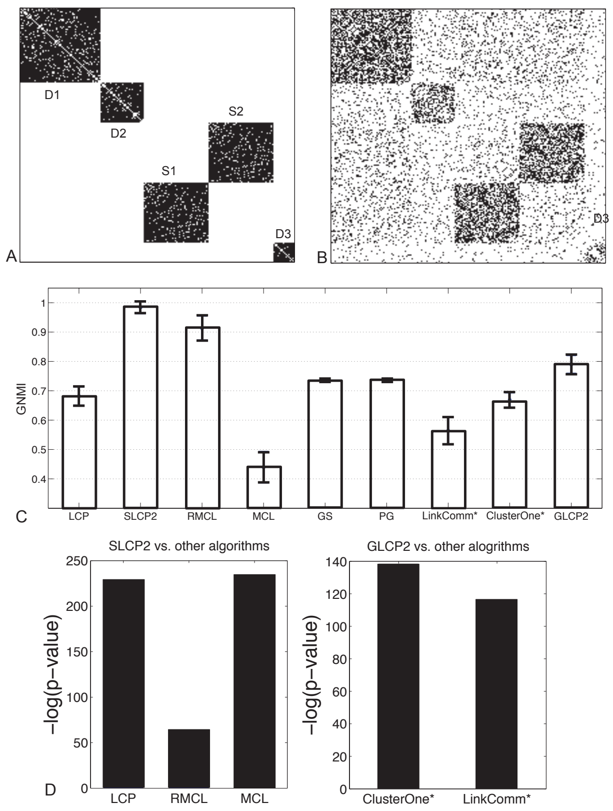


Figure 3.7: Performance comparison on synthetic networks: A. the adjacency matrix of the original network; B. one example of the randomly shuffled network (obtained by shuffling half of the original edges); C. GNMI comparison among all algorithms; D. t-test results.

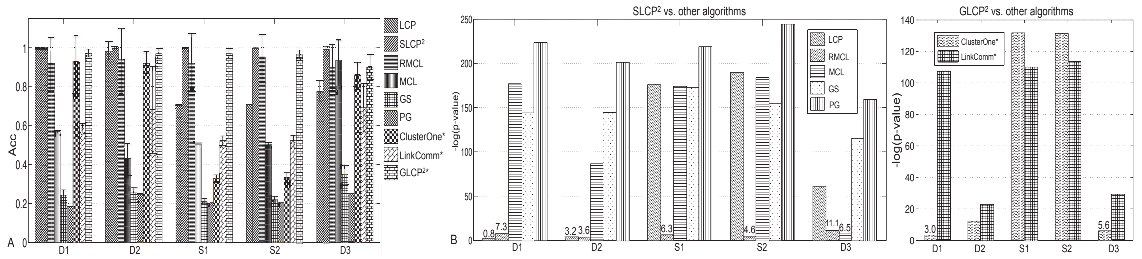


Figure 3.8: Statistical performance in detecting each module: A. *Acc* scores comparison among all algorithms; B. t-test results based on distributions of *Acc*. For low bars in B, we put the $-log(p - value)$ values on top of the bars.

Table 3.8: Performance comparison for complex prediction on *Sce* PPI networks.

Network	Dataset	Method	Coverage	#. clusters	#. matched	Sn	PPV	Acc
*Sce*DIP	MIPS	LCP(k=1000)	2572	525	62	0.2346	0.3825	0.2995
		RMCL	3725	814	79	0.2834	0.3977	0.3357
		MCL (I_F = 2.2)	3846	675	68	0.2821	0.3787	0.3269
		GS	2391	550	65	0.185	**0.4067**	0.2743
		PG	2717	364	14	0.1153	0.2978	0.1853
		SLCP^2(k=1000)	4564	783	84	0.3050	0.3732	0.3378
		ClusterOne*	1461	358	81	0.2641	0.3605	0.3085
		LinkComm*	2344	1725	102	**0.3093**	0.3575	0.3326
		GLCP^2*	3447	1517	**104**	0.3066	0.3928	**0.3470**
	SGD	LCP(k=1000)	2572	525	75	0.3484	0.6058	0.4594
		RMCL	3725	814	125	0.4572	0.6039	0.5254
		MCL (I_F=2.3)	3630	659	115	0.4468	0.5735	0.5102
		GS	2391	550	88	0.2915	**0.6689**	0.4416
		PG	2717	364	11	0.1714	0.4102	2615
		SLCP^2(k=1000)	4564	783	125	**0.4917**	0.5621	0.5257
		ClusterOne*	1461	358	113	0.4037	0.5775	0.4828
		LinkComm*	2344	1725	136	0.4567	0.4895	0.4727
		GLCP^2*	3447	1517	**155**	0.4894	0.5850	**0.5350**
*Sce*BG	MIPS	LCP(k=1000)	3503	557	77	0.2978	0.4252	0.3558
		RMCL	5210	772	81	0.4908	0.3921	0.4346
		MCL (I_F = 3.3)	3544	338	45	.3495	0.3270	0.3380
		GS	3315	609	83	0.2420	**0.4296**	0.3224
		PG	2601	356	2	0.0740	0.3128	0.1521
		SLCP^2(k=1000)	5209	782	84	0.3723	0.3906	0.3810
		ClusterOne*	2580	473	101	0.4797	0.3938	0.4346
		LinkComm*	4633	4108	**143**	**0.5891**	0.3526	0.4557
		GLCP^2*	4440	2183	136	0.5006	0.4204	**0.4587**
	SGD	LCP(k=1000)	3503	556	98	0.4672	0.6236	0.5398
		RMCL	5210	772	137	0.6628	0.5915	0.6262
		MCL (I_F = 3.2)	3652	335	80	0.4291	0.4752	0.4516
		GS	3315	609	130	0.3774	**0.6544**	0.4969
		PG	2601	356	3	0.135	0.4517	0.2469
		SLCP^2(k=1000)	5209	782	151	0.5847	0.5926	0.5886
		ClusterOne*	2580	473	158	0.6703	0.5621	0.6138
		LinkComm*	4633	4108	**207**	**0.7955**	0.4637	0.6037
		GLCP^2*	4440	2183	204	0.7341	0.5887	**0.6574**

Overlapping module identification algorithms are marked with a star *. *Sce*BG is short for *Sce*BioGrid.

Table 3.9: Performance comparison for complex prediction on *Hsa* PPI networks.

Network	Dataset	Method	Coverage	#. clusters	#. matched	Sn	PPV	Acc
*Hsa*HD	PCDq	LCP(k=1000)	8561	979	205	0.3986	0.4206	0.4095
		RMCL	6879	1508	290	0.3538	0.5990	0.4604
		MCL (I_F = 3.3)	6534	1279	237	0.3255	0.5633	0.4282
		GS	4719	1167	167	0.2169	**0.6785**	0.3836
		PG	5172	805	22	0.2016	0.3453	0.2639
		SLCP^2(k=1000)	8657	1494	303	0.3916	0.4774	0.4324
		ClusterOne*	2915	771	199	0.2379	0.6478	0.3925
		LinkComm*	7183	4107	418	**0.4314**	0.3029	0.3652
		GLCP^2*	8181	4257	**450**	0.4145	0.5377	**0.4721**
	CORUM	LCP(k=1000)	8561	979	172	0.3729	0.2049	0.2764
		RMCL	6879	1508	247	0.3291	0.2777	0.3023
		MCL (I_F = 3.3)	6534	1279	215	0.3192	0.2567	0.2862
		GS	4719	1167	195	0.2123	**0.3084**	0.2559
		PG	5172	805	2	0.1609	0.2084	0.1831
		SLCP^2(k=1000)	8657	1494	257	0.3748	0.2227	0.2889
		ClusterOne*	2915	771	233	0.2623	0.2624	0.2623
		LinkComm*	7183	4107	**614**	**0.4676**	0.1349	0.2510
		GLCP^2*	8181	4257	418	0.3859	0.2413	**0.3051**
*Hsa*BG	PCDq	LCP(k=1000)	7042	958	111	0.2798	0.4945	0.3720
		RMCL	10698	1536	223	0.3777	0.5054	0.4369
		MCL (I_F = 3.3)	5345	917	59	0.1668	**0.5563**	0.3046
		GS						
		PG						
		SLCP^2(k=1800)	12889	1622	205	0.3523	0.4281	0.3884
		ClusterOne*	10543	1753	162	0.4098	0.3869	0.3982
		LinkComm*	10322	6954	**372**	**0.4467**	0.2784	0.3526
		GLCP^2*	10948	5607	360	0.4190	0.4943	**0.4545**
	CORUM	LCP(k=1000)	958	7042	166	0.3558	0.2611	0.3047
		RMCL	10698	1536	190	0.4286	**0.2689**	0.3395
		MCL (I_F = 3.3)	5345	917	82	0.2094	0.2535	0.2304
		GS						
		PG						
		SLCP^2(k=1800)	12889	1622	221	0.4235	0.2331	0.3142
		ClusterOne*	10543	1753	197	0.5797	0.2548	0.3445
		LinkComm*	10322	6954	**724**	**0.6856**	0.1193	0.286
		GLCP^2*	10948	5607	615	0.5047	0.2313	**0.3476**

Overlapping module identification algorithms are marked with a star *. *Hsa*HD and *Hsa*BG are short for *Hsa*HPRD *Hsa*BioGrid, respectively. For *Hsa*BioGRID PPI network, GS and PG do not have results due to the memory limitation.

hour on a 2.4GHz quad-core CPU and 6GB RAM computer. Except when identifying modules in the *Hsa*BioGRID PPI network, PG and SG fail to execute due to the large memory requirement from two algorithms for this large PPI network. Based on the simulation results, the run time of SLCP^2 and GLCP^2 are very competitive with the other algorithms. For example, SLCP^2 only takes around two minutes for clustering the *Sce*DIP PPI network into $k = 1000$ modules and GLCP^2 needs less than one minute for analyzing the *Sce*DIP PPI network.

3.2.2.2 Synthetic networks

To illustrate the performance difference of different algorithms, we first evaluate all the selected algorithms on synthetic networks with the known ground truth. The modular structure of synthetic networks is shown in Fig. 3.7A. There are three dense modules of different sizes together with two sparse modules of the same size. In order to test statistical significance, we generate the null model by shuffling edges from an original synthetic network based on the Maslov-Sneppen procedure [60]. Fig. 3.7B is one example of the random network after half of the original edges are permuted. The performance is evaluated by Generalized Normalized Mutual Information (GNMI) [50] for both non-overlapping and overlapping module identification algorithms. GNMI ranges from 0 to 1 and it equals to 1 when the module identification result is the same as the ground truth.

Fig. 3.7C shows the mean values and the standard deviations of GNMI obtained by all the algorithms on 100 random null networks. For non-overlapping algorithms, SLCP^2 is superior to LCP, MCL and RMCL. For PG and GS, although the obtained GNMI values are better than LCP and MCL, they may not provide useful biological information as their identified modules are very fine grained (one or two nodes in each module). For overlapping module identification algorithms, GLCP^2 outperforms

ClusterOne and LinkComm. Fig. 3.7D plots the $-log(p - value)$ of the t-test scores of SLCP^2 compared to other non-overlapping algorithms as well as the comparison of GLCP^2 to ClusterOne and LinkComm. From Fig. 3.7D, we find that both SLCP^2 and GLCP^2 are significantly better than other state-of-the-art algorithms on synthetic networks with the ground truth modular structure.

In addition, we estimate the statistical significance for each identified module in synthetic random networks for all nine algorithms. We annotate the dense modules in Fig. 3.7A as D1, D2 and D3 and sparse modules as S1 and S2. Based on 100 random null networks, for each module, we can obtain the distribution of corresponding Acc scores based on the known ground truth. Fig. 3.8A displays the mean values and the standard deviations of Acc scores produced by all the algorithms on every module in Fig. 3.7A. For example, the first nine bars indicate the mean values and the standard deviations of Acc scores from all nine competing algorithms in detecting dense module D1 in Fig. 3.7A. Based on the distributions of Acc scores, we can further compute the p-values of our proposed algorithms compared to other state-of-the-art algorithms. Fig. 3.8B plots the $-log(p - value)$ of the t-test scores of SLCP^2 compared to other non-overlapping algorithms and the comparison of GLCP^2 to ClusterOne and LinkComm on all five modules, respectively. We consider our algorithms are significantly better when $-log(p - value) \geq 3$ ($p - value \leq 1.0e - 3$). From Fig. 3.8B, we find that LCP and SLCP^2 are competitive in identifying dense module D1. For the rest of the modules and algorithms, the $-log(p - value)$ values shown in Fig. 3.8B imply that our SLCP^2 and GLCP^2 achieve significantly better performance in detecting both dense and sparse modules. Furthermore, from Fig. 3.8B, we find the bars for sparse modules (S1 and S2) are typically higher than those corresponding to dense modules, which further validates that the competing algorithms focus more on detecting dense modules while our proposed algorithms can simultaneously detect both dense and sparse modules based on interaction patterns.

3.2.2.3 Complex prediction

We test the quality of a module identification algorithm by how well it can be applied to make predictions for protein complexes. We compare SLCP^2 with other state-of-the-art non-overlapping module identification algorithms, including LCP, RMCL, MCL, GS and PG, on four PPI networks. Also, to detect overlapping modules, we compare GLCP^2 with ClusterOne and LinkComm. The information of the module identification results and the optimal parameters used by each algorithm are reported in Table 3.8 and Table 3.9.

For non-overlapping module identification algorithms, as shown in Table 3.8 and Table 3.9, SLCP^2 and RMCL are competitive and outperform all the other non-overlapping algorithms. For the *Sce*DIP PPI network, SLCP^2 achieves better performance than RMCL because it predicts more matched protein complexes and has a higher *Acc* score. For other PPI networks, SLCP^2 and RMCL obtain competitive results as SLCP^2 consistently predicts more matched protein complexes while RMCL gets higher *Acc* scores. In addition, SLCP^2 has the best coverage with more proteins clustered into corresponding modules on all four PPI networks except the *Sce*BioGrid PPI network. In fact, for the *Sce*BioGrid PPI network, RMCL only covers one more protein than SLCP^2.

For overlapping module identification algorithms, based on Tables 3.8 and 3.9, we find that GLCP^2 outperforms LinkComm and ClusterOne. Although both GLCP^2 and LinkComm identify competitive numbers of protein complexes in different golden standards, GLCP^2 consistently achieves higher *Acc* scores for all four PPI networks. Finally, GLCP^2 also has the best coverage on all four PPI networks except the *Sce*BioGrid PPI network, on which LinkComm has a higher coverage than GLCP^2.

If we consider that LinkComm identifies larger numbers of smaller overlapping modules as shown in both tables, we expect that GLCP^2 may provide more biologically meaningful results.

Furthermore, we have tested the statistical significance of our algorithms in terms of predicting the SGD golden standard on the *Sce*DIP PPI network. We first generate 100 random networks from the original *Sce*DIP PPI network by randomly shuffling the original edges based on the Maslov-Sneppen procedure [60]. Then, we obtain the distributions of *Acc* scores with respect to the prediction of SGD golden standard on these 100 randomized networks for the competing algorithms. Based on the results provided in Table 3.8, we compare SLCP^2 with RMCL for non-overlapping algorithms and GLCP^2 with LinkComm for overlapping algorithms, because they are the two best-performing algorithms in predicting the SGD complexes among non-overlapping algorithms and overlapping algorithms, respectively. For non-overlapping algorithms, the average and the standard deviation of *Acc* scores obtained by SLCP^2 are 0.518 and 0.0064, respectively. While for RMCL, the average and the standard deviation of the *Acc* scores are 0.5137 and 0.0044, respectively. For overlapping algorithms, the average and the standard deviation of *Acc* scores of GLCP^2 are 0.5018 and 0.0054, respectively. For LinkComm, the average and the standard deviation of the *Acc* scores are 0.4983 and 0.0047, respectively. We calculate t-test scores based on these statistics and find that SLCP^2 is significantly better than RMCL with the p-value 2.59e-6 and GLCP^2 is significantly better than LinkComm with the p-value 1.05e-7.

In summary, both SLCP^2 and GLCP^2 based on our new optimization formulation LCP^2 using the concept of random walk on graphs are among the best performing algorithms for protein complex prediction. However, protein complexes have typical dense modular structure within which proteins are highly connected. As our SLCP^2 and GLCP^2 aim to detect both dense and sparse modules, these protein complex prediction results only exhibit one aspect of our algorithms' performance. In the following sections, we further compare the performance of different algorithms on functional module identification, especially for sparse module identification.

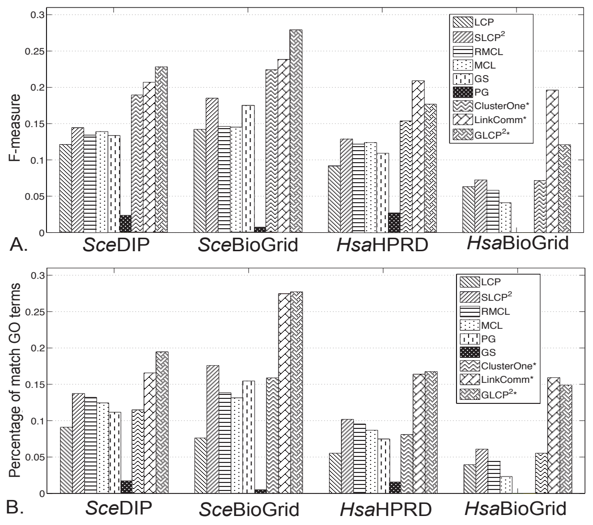


Figure 3.9: The top bar figure shows the comparison results based on the F measure on four PPI networks. The bottom figure displays the comparison of the percentages of matched GO terms in the complete set of selected high-level GO terms. For the HsaBioGRID PPI network, GS and PG fail to execute due to the memory limitation.

3.2.2.4 GO term prediction

In this section, we follow the same strategy in [96] to compare the biological significance of identified modules by all nine algorithms with respect to GO term prediction. Instead of using all GO terms, we only consider high level GO terms with information content larger than two so that we can better understand the functional specificity of identified modules. The comparison for GO term prediction is illustrated in Fig. 3.9. Figure 3.9A illustrates the F-measure comparison among all the algorithms. Figure 3.9B shows the percentage of GO terms that are considered to be correctly matched to at least one of the identified modules by different algorithms.

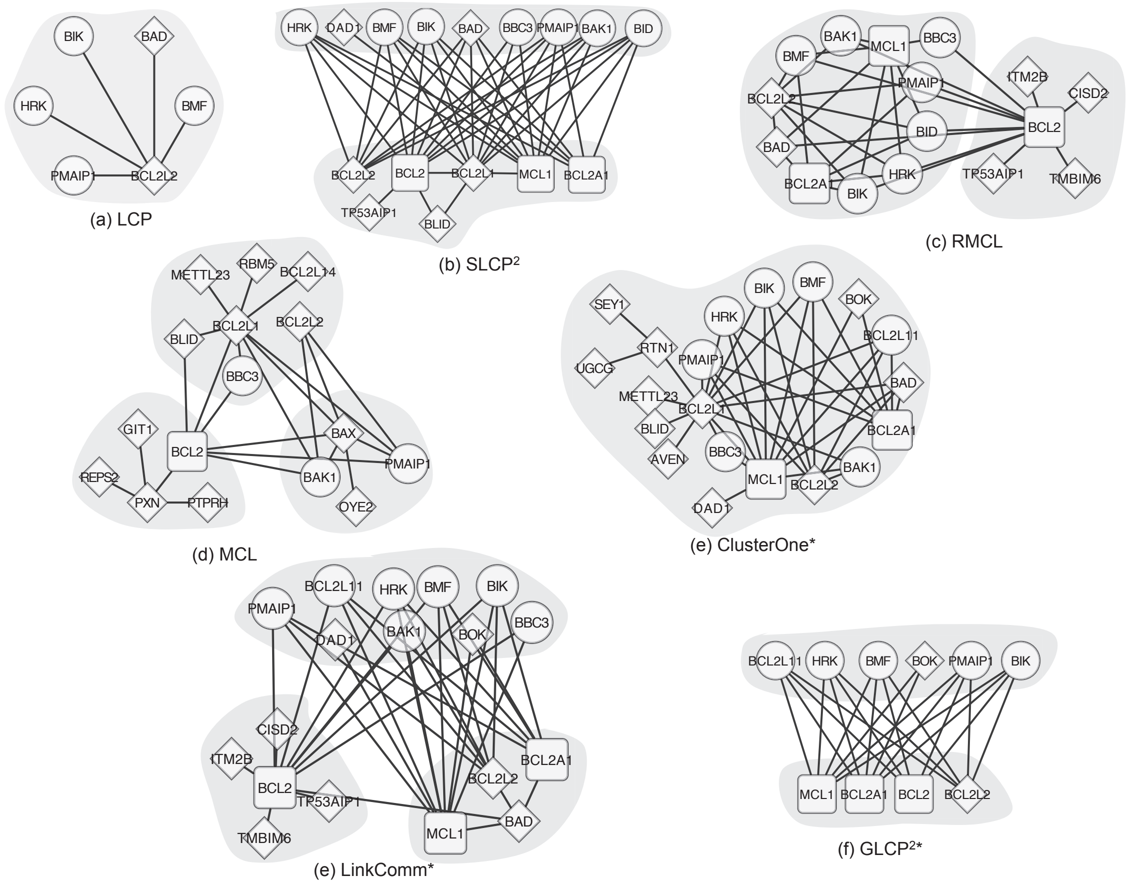


Figure 3.10: The pro-survival and cytochrome c release modules in *Hsa*BioGRID PPI network detected by all the algorithms (GS and PG fail to execute because of running out of memory). The pro-survial proteins are in rectangle shapes and the cytochrome c release proteins are in circle shapes. Diamond shapes denotes the proteins which belongs to neither the pro-survial proteins nor the cytochrome c release proteins. Shaded areas represent the modules detected by the algorithms.

Among non-overlapping algorithms, Fig. 3.9 clearly illustrates that SLCP^2 not only detects the largest number of matched high-level GO terms for each PPI network, but also obtains the best F-measure score. Therefore, for non-overlapping module identification, SLCP^2 outperforms other state-of-the-art non-overlapping algorithms on high level GO term prediction. For overlapping algorithms, based on Fig. 3.9, GLCP^2 identifies more matched GO terms and achieves higher F-measure scores than ClusterOne and LinkComm on two *Sce* PPI networks, which indicate that GLCP^2 outperforms ClusterOne and LinkComm for two yeast networks. For both *Hsa* PPI networks, GLCP^2 and LinkComm uncover competitive numbers of matched GO terms; however, LinkComm obtains better F-measure scores because it gets higher recall scores due to the fact that LinkComm detects a larger number of small overlapping modules since it does not have a post-processing procedure to deal with highly overlapping modules. These small overlapping modules can be matched to the same GO terms and hence the recall scores can get higher. Among all nine algorithms, for GO term prediction, GLCP^2 and LinkComm perform competitively with each other and outperform the other compared algorithms.

3.2.2.5 Sparse module identification

In order to further illustrate the advantage of our LCP^2 formulation in detecting functional modules with similar interaction patterns, we compare the performances of different algorithms with respect to identifying functional sparse modules in this section. However, in general, as we do not have sparse module golden standards, it is hard to provide quantitative measures for detecting sparse modules. In this section, we provide the examples of well understood biologically meaningful sparse modules to evaluate the capability of different algorithms in identifying functional sparse modules. Through the comparison of identified corresponding modules, we

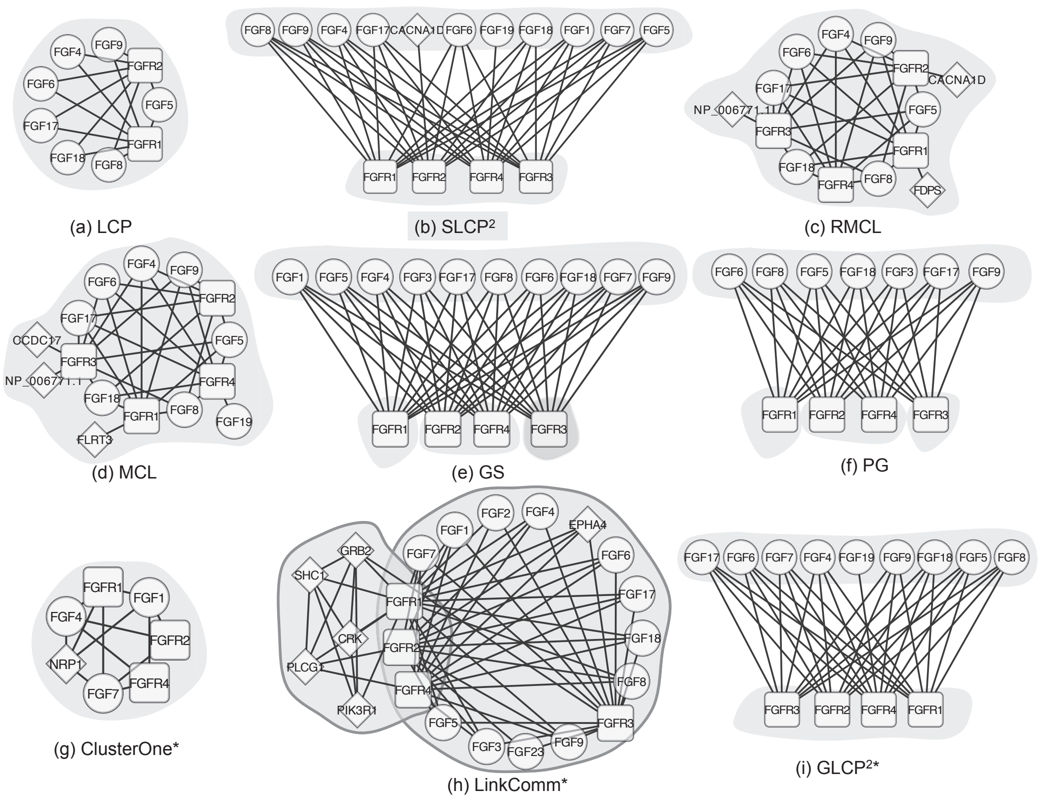


Figure 3.11: The FGF/FGFR signaling modules in *Hsa*HPRD PPI network detected by all algorithms. FGF proteins are in the circle shapes and FGFR proteins are in the rectangle shapes. Diamond shapes indicate proteins of neither FGF proteins nor FGFR proteins. Shaded areas represent the modules detected by the algorithms.

demonstrate that our SLCP^2 and GLCP^2 are superior in detecting functional sparse modules.

3.2.2.6 Pro-survival proteins and cytochrome c release

The pro-survival proteins (BCL2, MCL1 and BCL2A1), which constitute the Bcl-2 subfamily, directly or indirectly prevent the release of cytochrome c from mitochondria [117]. Therefore, the pro-survival proteins module should interact with the module which has the release of cytochrome c from mitochondria functionality. In Fig. 3.10, we provide the comparison of the module identification results for detecting these two modules in the *Hsa*BioGRID PPI network. For the pro-survival proteins module, we mark the three members in circle shapes. For functional module with the release of cytochrome c from mitochondria functionality (HRK, BCL2L11, BID, BNIP3, BIK, PMAIP1, BAK1, BMF and BBC3), we mark the members in rectangle shapes. Shaded areas represent the modules detected by the corresponding algorithms. Based on the interactions in the *Hsa*BioGRID PPI network, we find these two modules are two sparse modules within which proteins have similar interaction patterns. As shown in Fig. 3.10, LCP detects part of the cytochrome c release module but fails to identify the pro-survival module. RMCL splits pro-survival proteins into two modules. MCL fails to detect both the cytochrome c release module and the pro-survival module. ClusterOne groups those two modules into one. LinkComm fails to detect the pro-survival modules. Only our algorithms SLCP^2 and GLCP^2, which take the interaction patterns into account, achieve the most promising results. For two algorithms PG and GS, which also consider the interaction patterns, we do not have their module detection results because both algorithms run out of memory on this relatively large network.

3.2.2.7 FGF/FGFR signaling

FGF/FGFR signaling has been associated with a diverse and broad range of biological functions, including cell growth, cell differentiation, and the promotion of angiogenesis [81]. FGFR stands for the fibroblast growth factor receptors, which bind to the members of the family of FGF (fibroblast growth factor) proteins. Based on their functionality, FGFR proteins should interact with FGF proteins. Fig. 3.11 illustrates the module identification results for FGFR and FGF modules in the *Hsa*HPRD PPI network. Based on the network structure, FGFR and FGF modules are two sparse modules. We mark the FGFR proteins in rectangle shapes and FGF proteins in circle shapes. Shaded areas represent the modules detected by the corresponding algorithms. As shown in Fig. 3.11, LCP, RMCL, MCL, ClusterOne and LinkComm again can not identify these two modules correctly. PG, GS and our algorithms have the ability to correctly detect them. However, PG and GS over-segment the FGFR module while our algorithms can provide better module identification results.

3.2.3 Discussion and conclusions

The compared module identification algorithms in this section use different module definitions and methods. LCP, ClusterOne, SLCP^2, and GLCP^2 are all based on finding low conductance sets defined by the Markov chain of random walk on networks. LCP and ClusterOne are the non-overlapping and overlapping algorithms of searching for low conductance sets defined by the transition matrix P (LCP) of the underlying Markov chain. Therefore, they tend to find densely connected modules. However, SLCP^2 and GLCP^2 are respective algorithms for searching for non-overlapping and overlapping modules by finding low conductance sets based on the two-hop transition matrix P^2 (LCP^2) of the random walk Markov chain. By taking the advantage of finding two-hop low conductance sets, our new algorithms detect modules based on the nodes interaction patterns, which reflect functional similarity between proteins. In [92], the authors present a similar formulation to search for modules based on the interaction similarity. However, our formulation depending on the Gram matrix W derived by LCP^2 can be viewed as the normalized version of the symmetrization matrix proposed in [92]. Generally, as in normalized cuts, the normalized version often gives balanced modules that may lead to more promising functional module identification results. Both MCL and RMCL are network clustering algorithms based on (stochastic) flow simulation which extends the similar random walk Markov chain idea by two operations for better performance: "Inflation" and "Expand". However, both operators are heuristic strategies. Theoretically, why they give good results is still a mystery. PG and SG are two non-overlapping algorithms that identify functional modules in terms of interaction patterns. Because they apply greedy algorithms to solve the module identification problem, the optimal quality of the results is not guaranteed. Last but not least, LinkComm is a novel overlapping algorithm based on an edge graph representation that tends to detect a large number of overlapping modules whose biological meaning may not be immediately clear due to the fine grained modular structure.

In our experiments, we have applied our algorithms to analyze four unweighted PPI networks, which can be viewed as binary ($\{0, 1\}$) edge-weighted networks. However, both SLCP^2 and GLCP^2 can be extended in a straightforward manner for the analysis of general edge-weighted networks by modifying corresponding terms in Algorithms 1 and 2 proposed in this section. We will evaluate the performances of algorithms in module identification by introducing reliable edge weights when they are available in our future work. Another limitation for SLCP^2 is how to decide the desirable number of modules k in advance. One possible way is to search k values within a certain range and choose k with the best average weight density computed by (3.16). In our future research, we will also explore the ideas adopted in [1] and [18] to determine k based on the partition density and/or module entropy score, respectively. Finally, GLCP^2 is our preliminary solution strategy for identifying overlapping modules based on the LCP^2 formulation. We plan to further investigate the properties of the Gram matrix W and we expect that we may achieve better performance with a better understanding of the problem structure.

In conclusion, we propose a novel formulation to achieve functional module identification based on protein interaction patterns in PPI networks. An efficient spectral algorithm, which can obtain a close-to-optimal solution based on $Ky\ Fan\ theorem$, is designed to solve the new optimization problem for non-overlapping module identification. We also develop a greedy algorithm to solve the same problem but obtain overlapping results. Our algorithms not only can overcome the limitation of traditional module identification algorithms, which only focus on identifying dense modules, but they also have a better scalability for large-scale PPI networks to efficiently solve module identification problem. Experimental results show that our SLCP^2 and GLCP^2 have achieved promising results on both protein complex and GO term predictions on four large-scale PPI networks. Most importantly, our new algorithms can detect functional sparse modules, which are often ignored by many other existing algorithms.

3.3 Non-negative matrix factorization framework

In this section, we propose a flexible NMF based formulation to identify functional modules based on block modeling. We briefly review the related work in section 3.3.1, followed by the derivation of our novel formulation in section 3.3.2 and the alternating proximal method (APANMF) in Section 3.3.3. The convergence-related propositions of our APANMF (Propositions 1, 2 and 3) are also provided in section 3.3.3. In sec-

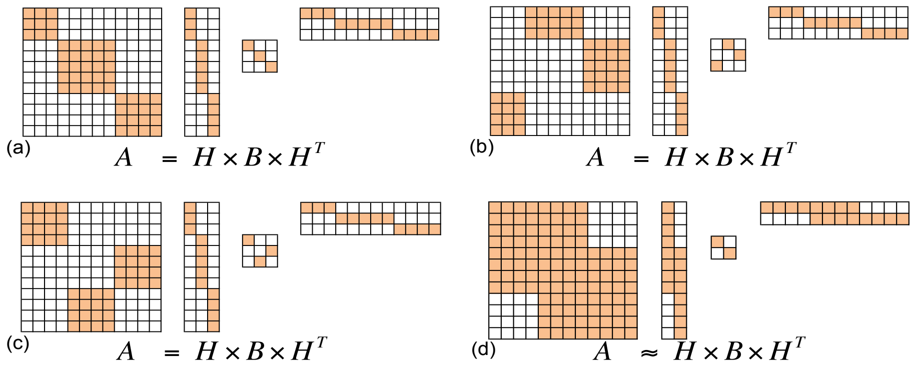


Figure 3.12: Graph clustering under different settings: (a) Toy example for community detection for undirected graph. (b) Toy example for directed graph clustering. (c) Toy example for block modeling clustering. (d) Toy example for overlapping graph clustering.

tion 3.3.4, we demonstrate the superiority of our APANMF by comparing with other state-of-the-art methods (SymNMF_MU [24], SymNMF_NT [47], ASymNMF [106], BNMF [19]) on synthetic networks (LFR benchmarks [49] and block modeling benchmarks [109]) as well as real-world large-scale network datasets (Facebook ego network from http://snap.stanford.edu/data/ and PIPs human protein-protein interaction (PPI) network [62]). We draw the conclusion in section 3.3.5.

3.3.1 Related work

The authors in [47, 24] propose to decompose the adjacency matrix A of network G into symmetric components for community detection:

$$\min_{X \geq 0}: \Gamma(X) = \left\| A - XX^T \right\|_F^2, \tag{3.17}$$

where X is a non-negative matrix of size $n \times K$ and K is the number of potential modules. X can be naturally interpreted as the module assignment matrix. A multiplicative updating algorithm SymNMF_MU [24] has been proposed to solve this problem (3.17). However, SymNMF_MU may not converge to a stationary point, which will be further discussed in Section 3.3.3.3. SymNMF_NT [47] is a Newton-like algorithm, which solves the problem (3.17) by lining up the columns of X. SymNMF_NT converges to a stationary point. However, it has relatively larger memory consumption requirement [47].

In order to handle directed graphs, the authors in [106] have presented an asymmetric NMF decomposition formulation:

$$\min_{X\geq 0,\ S\geq 0}: \Pi(X,S) = \left\|A - XSX^T\right\|_F^2, \tag{3.18}$$

where $S_{K\times K}$ is a $K \times K$ asymmetric matrix for handling the asymmetric adjacency matrix A of a network with directed edges. A multiplicative updating algorithm ASymNMF [106] has been developed to solve this problem (3.18). The objective function values generated by ASymNMF monotonically decrease but the solution may not converge to a stationary point, which is discussed in section 3.3.3.3.

For block modeling graph clustering, one recent algorithm—BNMF [19]—has been derived base on the following formulation:

$$\min_{X\geq 0,\ 0\leq M\leq 1}: \left\|A - XMX^T\right\|_F^2 + \lambda\left\|M^{ideal} - M\right\|_F^2, \tag{3.19}$$

where M and M^{ideal} represent the adjacency matrices of the introduced image graph and the "ideal image matrix", respectively. M^{ideal} is the function of M, which is defined by $M_{ij}^{ideal} = \underset{u\in\{0,1\}}{argmin}\,|u - M_{ij}|$ and approximated by a sigmoid function in the proposed projected descent algorithm. However, there is no convergence proof provided for BNMF.

3.3.2 Flexible graph clustering with L1-norm regularization

Adopting different NMF-based formulations can address different network partition problems, such as aforementioned community detection and block modeling for networks with either undirected or directed edges, by different formulations (3.17), (3.18), and (3.19). In this section, we propose a mathematical formulation, which can deal with all the above tasks in just one flexible framework. Furthermore, we explicitly control the sparsity of factorized components by adding L1-norm penalty terms to yield sparse and robust solutions for noisy networks.

3.3.2.1 A flexible graph clustering formulation

Our formulation is based on the similar assumption that the given adjacency matrix can be factorized by the multiplications of a module assignment matrix and an adjacency matrix of the image graph capturing the underlying topology of the given graph $A \approx XBX^T$ [36]:

$$\begin{aligned} \text{min: } & \left\| A - XBX^T \right\|_F^2, \\ s.t.\ & X_{ij} \in \{0,1\}, \forall i,j; \\ & B_{rs} \in \{0,1\}, \forall r,s, \end{aligned} \tag{3.20}$$

where X is the $n \times K$ dimensional assignment matrix with $X_{ir} = 1$ revealing that vertex i belongs to cluster r and $X_{ir} = 0$ otherwise. The introduced image graph is presented by the adjacency matrix B, in which B_{rs} indicates the connectivity between the cluster r and the cluster s with $B_{rs} = 1$ meaning that cluster r densely interacts with the cluster s and $B_{rs} = 0$ otherwise. We note that our formulation is similar to (3.18), but with the binary constraints on both X and B. Detecting modules by our formulation may provide better physical interpretations for both the assignment matrix X and the image graph B.

By solving the optimization problem (3.20), we can obtain the promising graph clustering results. However, it is challenging to find integer solutions for this nonlinear optimization problem (3.20) due to the inherent NP hardness of general network clustering as a quadratic assignment problem [109, 111], especially with large-scale networks. Relaxing the constraints from integer to continuous variables is one typical way to achieve high quality solutions [29]. In this section, we relax our binary constraints as follows:

$$\varphi = \left\{(X, B)|0 \leq X_{ij} \leq 1, 0 \leq B_{rs} \leq 1, \forall i, j, r, s\right\}. \tag{3.21}$$

The relaxed search space φ allows the elements in X and B range from 0 to 1. After relaxation (3.21), our problem becomes:

$$\min_{(X,B)\in\varphi} : \Psi(X, B) = \left\|A - XBX^T\right\|_F^2. \tag{3.22}$$

3.3.2.2 L1-norm regularization

For noise free networks, such as toy examples given in Fig. 3.12, or networks with reasonably low noise, our proposed formulation (3.22) can naturally produce sparse results with original clustering structures because the assumption $A \approx XBX^T$ holds. However, for real-world networks, which often contain significant amount of noise due to limitations of interaction profiling methods, the underlying clustering structures may be destroyed and the assumption $A \approx XBX^T$ may not be satisfied. Hence, we may not be able to have meaningful sparse results by directly solving (3.22). In order to address this problem, we add L1-norm regularization terms for both X and

B to (3.22) to explicitly enforce sparse structures for X and B:

$$\min_{(X,B)\in\varphi} : \Omega(X,B) = \left\| A - XBX^T \right\|_F^2 + \alpha \left\| X \right\|_{L1} + \beta \left\| B \right\|_{L1}, \tag{3.23}$$

in which $\|X\|_{L1} = \sum_{i,j} |X_{ij}|$. With the newly added regularization terms, we hope for the guarantee of physically meaningful sparse results, especially for noisy networks.

3.3.3 Alternating proximal algorithm

To solve this sparse NMF-based graph clustering problem, we now derive a new set of optimization algorithms, which are different from the existing algorithms, mostly based on multiplicative updating algorithms for the original NMF algorithm [52]. Mathematically, our optimization problem (3.23) is more challenging to solve with two non-differentiable terms in $\Omega(X,B)$, compared to the optimization problems (3.17), (3.18) and (3.19). In order to efficiently solve this optimization problem (3.23), we need to make use of the structure of the objective function, which takes the sum of a differentiable component and other non-differentiable components. Based on this observation, we develop an alternating proximal method that optimizes the cluster assignment matrix X and the image matrix B in an alternating way. This alternating proximal algorithm is guaranteed to converge to a stationary point of the optimization problem (3.23).

3.3.3.1 Updating X

Let us first consider the optimization step with respect to the assignment matrix X by fixing the image matrix at $\hat{B}$. The decomposed optimization problem aims to solve the following problem:

$$\min_{0\leq X\leq 1} : F(X) = \left\| A - X\hat{B}X^T \right\|_F^2 + \alpha \left\| X \right\|_{L1}, \tag{3.24}$$

where we define $P(X) = \left\| A - X\hat{B}X^T \right\|_F^2$ and $P(X)$ is differentiable.

Because of the structure of the problem, we apply a proximal method to iteratively solve the optimization problem. As similarly done in [58], we propose to compute $G_k(X)$ for the approximation of $F(X)$ at the kth iteration around X^{k-1}:

$$\begin{aligned} G_k(X) = P(X^{k-1}) + &< \nabla P(X^{k-1}), (X - X^{k-1}) > \\ &+ \frac{L_k}{2} \left\| X - X^{k-1} \right\|_F^2 + \alpha \left\| X \right\|_{L1}, \end{aligned} \tag{3.25}$$

where L_k is a Lipschitz constant, which can be chosen to satisfy the following inequality:

$$G_k(X^k) \geq F(X^k). \tag{3.26}$$

Hence, instead of finding X^k based on $F(X^{k-1})$, our proximal method solves the following problem at the kth iteration:

$$X^k = \underset{0 \leq X \leq 1}{\arg\min} : G_k(X). \tag{3.27}$$

After some algebraic manipulations by completing the square and removing the constant terms, the problem (3.27) is in fact equivalent to the following problem:

$$\begin{aligned} X^k = &\underset{0 \leq X \leq 1}{\arg\min} \\ &\left\{ \alpha \left\| X \right\|_{L1} + \frac{L_k}{2} \left\| X - \left(X^{k-1} - \frac{1}{L_k} \nabla P(X^{k-1}) \right) \right\|_F^2 \right\}. \end{aligned} \tag{3.28}$$

Furthermore, we notice that this equivalent problem (3.28) has a closed-form solution, which is a promising property of our proximal method. With the closed-form solution, we can efficiently solve (3.28) without intensive computation. The closed-from solution is provided in Proposition 1, whose proof is given in the appendix.

Algorithm 1 Proximal Method for updating X (PMH($X, \hat{B}$))

```
1. Input: $X^0$, $\hat{B}$, $k = 1$, $L_0 > 1$, $\eta > 0$ and $\xi > 0$;
2. Output: $X^*$;
3. do
4.    Find the smallest non-negative integer $i_k$ such that
inequality (3.26) is satisfied with $L_k = \eta^{i_k} L_{k-1}$ ;
5.    Obtain $X^k$ from (3.30);
6.    $k = k + 1$;
7. while$\big(F(X^{k-1}) - F(X^k) > \xi\big)$
8. $X^* = X^k$.
```

Proposition 1. *For the following optimization problem:*

$$X^k = \arg \min_{0 \leq X \leq 1} \left\{ \phi(X) = \alpha \left\| X \right\|_{L1} + \frac{L_k}{2} \left\| X - \bar{X} \right\|_F^2 \right\}, \tag{3.29}$$

where $\bar{X} = X^{k-1} - \frac{1}{L_k} \nabla P(X^{k-1})$, the element-wise closed-form solution is

$$X_{ij}^k = \mathbb{P}(\mathbf{prox}_X(\bar{X})_{ij}), \tag{3.30}$$

where $\mathbb{P}(\cdot)$ is the projection operator and it is defined by

$$\mathbb{P}(x) = \begin{cases} 1 & x > 1 \\ x & 0 \leq x \leq 1 \\ 0 & x < 0 \end{cases}, \tag{3.31}$$

and

$$\begin{aligned} & \mathbf{prox}_X(\bar{X}) \\ & = \arg \min_X \left\{ \phi(X) = \alpha \left\| X \right\|_{L1} + \frac{L_k}{2} \left\| X - \bar{X} \right\|_F^2 \right\}, \end{aligned} \tag{3.32}$$

whose result is the solution of $\frac{\partial \phi(X)}{\partial X} \ni \boldsymbol{0}$ and can be computed in the following equa-

tion:

$$\mathbf{prox}_X(\bar{X})_{ij} = \begin{cases} 0 & |\bar{X}_{ij}| \le \frac{\alpha}{L_k} \\ \bar{X}_{ij} - \frac{\alpha}{L_k}\mathrm{sign}(\bar{X}_{ij}) & |\bar{X}_{ij}| > \frac{\alpha}{L_k} \end{cases} \tag{3.33}$$

The proximal method for updating X (PMH) is described in Algorithm 1. The convergence of PMH is guaranteed by Proposition 2 with the proof given in the appendix.

Proposition 2. *The sequence $\{F(X^k)\}_{k\ge 0}$ generated by the algorithm in Algorithm 1 monotonically decreases and the sequence $\{S_k(X^k) = G_k(X^k) - F(X^k)\}_{k\ge 0}$ converges to zero. Furthermore, when $k \mapsto +\infty$, X^k satisfies an asymptotic stationary point condition.*

Proof. We can prove the fact that $\{F(X^k)\}_{k\ge 0}$ is non-increasing and convergent due to the following inequalities:

$$F(X^k) \le G_k(X^k) \le G_k(X^{k-1}) = F(X^{k-1}). \tag{3.34}$$

The first inequality comes from the fact that $G_k(X)$ is the upper bound of $F(X)$ (3.26). We have the second inequality as the proximal method solves (3.27). The last equality can be obtained by substituting X with X^{k-1} in (3.25). Because $\{F(X^k)\}_{k\ge 0}$ is bounded, we define F^* as its limit. Based on (3.34) and $S_k(X^k) = G_k(X^k) - F(X^k)$, we have:

$$S_k(X^k) \le F(X^{k-1}) - F(X^k). \tag{3.35}$$

By adding all the terms over k, we have

$$\sum_k S_k(X^k) \le F(X^0) - F^*, \tag{3.36}$$

which is also bounded. Therefore, $\left\{S_k(X^k)\right\}_{k\geq 0}$ necessarily converges to zero.

Furthermore, we notice that $S_k(X)$ is differentiable and Lipschitz continuous because

$$\begin{aligned} S_k(X) =& G_k(X) - F(X) \\ =& P(X^{k-1}) - P(X) + \frac{L_k}{2}\left\|X - X^{k-1}\right\|_F^2 \\ &+ < \nabla P(X^{k-1}), (X - X^{k-1}) > . \end{aligned} \tag{3.37}$$

Therefore, for any X^k and X', $S_k(X)$ satisfies the classical lemma (lemma 1.2.3 in [70]), which yields

$$S_k(X') \leq S_k(X^k) - \frac{1}{2L_k}\left\|\nabla S_k(X^k)\right\|_F^2, \tag{3.38}$$

where we define $X' = X^k - \frac{1}{L_k}\nabla S_k(X^k)$. Here we make a mild assumption that both X' and X^k are in the constraint set. A similar assumption has been made for proving the convergence of a constrained optimization problem [58]. From (3.38), we derive

$$\begin{aligned} \left\|\nabla S_k(X^k)\right\|_F^2 &\leq 2L_k(S_k(X^k) - S_k(X')) \\ &\leq 2L_k S_k(X^k) \overset{k\mapsto+\infty}{\mapsto} 0, \end{aligned} \tag{3.39}$$

where we take the fact that $S_k(X') \geq 0$ because (3.34) and $\left\{S_k(X^k)\right\}_{k\geq 0}$ converges to zero.

Now, we compute the directional derivative $\nabla_{X-X^k}F(X^k)$ of $F(\cdot)$ at X^k in the direction $X - X^k$,

$$\begin{aligned} \nabla_{X-X^k}F(X^k) =& \nabla_{X-X^k}G_k(X^k) \\ &- < \nabla S_k(X^k), X - X^k > . \end{aligned} \tag{3.40}$$

Note that X^k minimizes G_k on $\{X|0 \leq X \leq 1\}$ and therefore $\nabla_{X-X^k} G_k(X^k) \geq 0$ [14]. With these,

$$\nabla_{X-X^k} F(X^k) \geq - \left\| \nabla S_k(X^k) \right\|_F \left\| X - X^k \right\|_F , \tag{3.41}$$

based on Cauchy-Schwarz inequality. Then,

$$\lim_{k \mapsto +\infty} \frac{\nabla_{X-X^k} F(X^k)}{\left\| X - X^k \right\|_F} \quad \geq \quad \lim_{k \mapsto +\infty} - \left\| \nabla S_k(X^k) \right\|_F = 0, \tag{3.42}$$

which further indicates that X^k is the stationary point of $F(X)$ when k approaches $+\infty$ based on the definition of an asymptotic stationary point proposed in [58].

□

Algorithm 2 Proximal Method for updating B (PMB($\hat{X}, B$))

```
1. Input: $\hat{X}$, $B^0$, $k = 1$ and $\xi > 0$;
2. Output: $B^*$;
3. do
4.    Compute $U_k(B)$ based on (3.47);
5.    Compute $B^k$ based on (3.49);
6.    $k = k + 1$
7. while($E(B^{k-1}) - E(B^k) > \xi$)
8. $B^* = B_k$.
```

3.3.3.2 Updating B

Updating B is similar as updating X because the optimization with B has the same structure as (3.24). Given an assignment matrix $\hat{X}$, the optimization problem we want to solve is:

$$\min_{0 \leq B \leq 1} : \; E(B) = \left\| A - \hat{X} B (\hat{X})^T \right\|_F^2 + \beta \left\| B \right\|_{L1}, \tag{3.43}$$

where $\|B\|_{L1}$ is the non-smooth term while $\Phi(B) = \left\| A - \hat{X} B (\hat{X})^T \right\|_F^2$ is differentiable with the gradient $\nabla \Phi(B) = 2((\hat{X})^T \hat{X} B (\hat{X})^T \hat{X} - (\hat{X})^T A \hat{X})$. Here, the square of the largest eigenvalue of $(\hat{X})^T \hat{X}$ is $\Phi(B)$'s Lipschitz constant L_B, which can be proven with Lemma 1.

Lemma 1. $\Phi(B) = \left\| A - X B X^T \right\|_F^2$ *is Lipschitz continuous and its Lipschitz constant* Π *is equal to the square of the largest eigenvalue of* $X^T X$ *(*$L_B = \delta_{max}^2 (X^T X)$*).*

Proof. Given two matrices X and Y, we have

$$\begin{aligned} & \left\| \nabla \Phi(X) - \nabla \Phi(Y) \right\|_F^2 \\ & = \left\| X^T X (X - Y) X^T X \right\|_{F2}^2 \\ & = trace(X^T X (X - Y)^T X^T X X^T X (X - Y) X^T X), \end{aligned} \tag{3.44}$$

where $X^T X$ is a positive semi-definite symmetric matrix. Hence, we can write $X^T X = U \Sigma U^T$ by SVD (singular value decomposition) with $U U^T = I_n$ and $U^T U = I_k$. By straightforward algebraic manipulations, (3.44) is equivalent to

$$\begin{aligned} & \left\| \nabla \Phi(X) - \nabla \Phi(Y) \right\|_F^2 \\ & = trace(U \Sigma U^T (X - Y)^T U \Sigma U^T U \Sigma U^T (X - Y) U \Sigma U^T) \\ & = trace(U^T (X - Y)^T U \Sigma^2 U^T (X - Y) U \Sigma^2) \\ & \leq \delta_{max}^4 trace(U^T (X - Y)^T U U^T (X - Y) U) \\ & = \delta_{max}^4 \left\| X - Y \right\|_F^2, \end{aligned} \tag{3.45}$$

where δ_{max} is the largest eigenvalue of $X^T X$. From (3.45), we have the following inequality:

$$\left\|\nabla\Phi(X) - \nabla\Phi(Y)\right\|_F \leq \delta_{max}^2 \left\|X - Y\right\|_F . \tag{3.46}$$

Therefore, $\Phi(B)$ is Lipschitz continuous and the Lipschitz constant L_B is equal to the square of the largest eigenvalue of $X^T X$. $\square$

We adopt a similar proximal method by approximating $E(B)$ in (3.43) at B^{k-1} by an upper-bound function:

$$\begin{aligned} U_k(B) &= \Phi(B^{k-1}) + < \nabla\Phi(B^{k-1}), (B - B^{k-1}) > \\ &+ \frac{L_B}{2}\left\|B - B^{k-1}\right\|_F^2 + \beta \left\|B\right\|_{L1} \\ &= \beta \left\|B\right\|_{L1} + \frac{L_B}{2}\left\|B - \bar{B}\right\|_F^2 , \end{aligned} \tag{3.47}$$

where $\bar{B} = B^{k-1} - \frac{1}{L_B}\nabla\Phi(B^{k-1})$. At the kth iteration, we solve the optimization problem:

$$B^k = \underset{0 \leq B \leq 1}{\arg\min} : U_k(B). \tag{3.48}$$

The corresponding closed-form optimal solution is derived similarly as Proposition 1

$$B_{ij}^k = \mathbb{P}(\mathbf{prox}_B(\bar{B})_{ij}), \tag{3.49}$$

where

$$\mathbf{prox}_B(\bar{B})_{ij} = \begin{cases} 0 & |\bar{B}_{ij}| \leq \frac{\beta}{L_B} \\ \bar{B}_{ij} - \frac{\beta}{L_B}\mathrm{sign}(\bar{B}_{ij}) & |\bar{B}_{ij}| > \frac{\beta}{L_B} \end{cases} , \tag{3.50}$$

Algorithm 4.2 details the procedure of the proximal method for updating B (PMB). We note that $E(B)$ is convex with respect to B and the constraint set $0 \leq B \leq 1$ is also convex. Therefore, the algorithm (PMB) converges to an optimal solution for a fixed $\hat{X}$ [10].

3.3.3.3 Alternating proximal algorithm for NMF (APANMF)

With both the algorithms PMH and PMB in hands, we summarize the alternating proximal algorithm (APANMF) in Algorithm 3. The convergence of APANMF is guaranteed by Proposition 3, whose proof is provided in the appendix.

Proposition 3. *The sequence of* $\{\Omega(X^t, B^t)\}_{t\geq 0}$ *monotonically decreases*

$$\Omega(X^{t+1}, B^{t+1}) \leq \Omega(X^t, B^t). \tag{3.51}$$

Furthermore, the sequence $\{(X^t, B^t)\}_{t\geq 0}$ *converges to an asymptotic stationary point.*

Proof. At the tth iteration, we have $\Omega(X^t, B^t)$. Based on Proposition 2 (3.34) for a fixed B^t, we get

$$\Omega(X^{t+1}, B^t) \leq \Omega(X^t, B^t). \tag{3.52}$$

Furthermore, based on (3.42), we have

$$\frac{\nabla_{X-X^{t+1}} F(X^{t+1})}{\|X - X^{t+1}\|_F} \geq 0 \Leftrightarrow \frac{\nabla_{Q-Q^{t+1,t}} \Omega(Q^{t+1,t})}{\|Q - Q^{t+1,t}\|_F} \geq 0, \tag{3.53}$$

where X^{t+1} is an asymptotic stationary point and we define $Q^{t+1,t} = [X^{t+1}; B^t]$.

Similarly, the proof in [8] demonstrates that for a fixed X^{t+1} we can obtain B^{t+1} satisfying

$$\Omega(X^{t+1}, B^{t+1}) \leq \Omega(X^{t+1}, B^t) \tag{3.54}$$

as $\Omega(X^{t+1}, B^t)$ is convex with respect to B^t for the given X^{t+1}. Similar to (3.53), for the asymptotic stationary point B^{t+1}, we have

$$\frac{\nabla_{B-B^{t+1}} E(B^{t+1})}{\|B - B^{t+1}\|_F} \geq 0 \Leftrightarrow \frac{\nabla_{Q-Q^{t+1,t+1}} \Omega(Q^{t+1,t+1})}{\|Q - Q^{t+1,t+1}\|_F} \geq 0. \tag{3.55}$$

From (3.52) and (3.54), we know

$$\Omega(X^{t+1}, B^{t+1}) \le \Omega(X^{t+1}, B^{t}) \le \Omega(X^{t}, B^{t}) \tag{3.56}$$

Obviously, the sequence $\{(X^t, B^t)\}$ is non-increasing and bounded. We further assume that $\tilde{Q} = [\tilde{X}; \tilde{B}]$ is a limit point of the sequence. Based on (3.53) and (3.55), for any Q in the constraint set, we obtain

$$\frac{\nabla_{Q-\tilde{Q}}\Omega(\tilde{Q})}{\left\|Q-\tilde{Q}\right\|_F} = \lim_{t\mapsto+\infty} \frac{\nabla_{Q-Q^{t,t}}\Omega(Q^{t,t})}{\|Q-Q^{t,t}\|_F} \ge 0, \tag{3.57}$$

which means that $\tilde{Q}$ is an asymptotic stationary point of $\Omega(Q)$ [58]. □

Algorithm 3 Alternating Proximal Algorithm

```
1. Input: $A_{n\times n}$ and $K$;
2. Output: $X$ and $B$;
3. Initialization: $X^0_{n\times K} > 0$, $B^0_{K\times K} > 0$ and $t = 1$;
4. do
5.    $X^{t+1}$ =PMH($X^t, B^t$);
6.    $B^{t+1}$ =PMB($X^{t+1}, B^t$);
7.    $t = t + 1$;
8. while($\Omega(X^{t-1}, B^{t-1}) - \Omega(X^t, B^t) > \xi$)
9. Compute $X$ by normalizing each row of $X^t$ to have the unit length.
```

One profound contribution of our APANMF is that APANMF has the theoretical guarantee to converge to a stationary point, which neither SymNMF_MU nor ASymNMF has provided. Additionally, to the best of our knowledge, this is the first convergence proof of a coordinate descent method for solving the NMF problem, one of whose decomposed optimization problems is non-convex and non-smooth. Therefore, our proof could provide insightful guidance for the convergence proof of the

NMF problems with similar structures. For SymNMF in general settings, the stationary points of the optimization problem in (1) necessarily contain zero elements: $\exists i, j,\ X^*_{ij} = 0$ (Proposition 4 in the appendix). Meanwhile, the proposed multiplicative algorithm SymNMF [47] always generates iterative updates in the positive orthant (Proposition 6): $X^k_{ij} > 0, \forall i, j$. Therefore, SymNMF may not converge in general when $\exists i, j,\ X^*_{ij} = 0$. Similarly for ASymNMF [106], although the authors have shown that the sequence of objective function values during the iterative procedure of ASymNMF monotonically decreases, it is not enough to say that ASymNMF converges to a stationary point. Specifically, Proposition 7 shows that the algorithm updates in the positive orthants for both X and C ($X^k_{ij} > 0, C^k_{rs} > 0,\ \forall i, j, r, s$) while Proposition 5 indicates that the stationary points contain zero elements in general ($\exists i, j, r, s,\ X^*_{ij} = 0$ or $C^*_{rs} = 0$ in the stationary point). HXence, no convergence properties of the sequences $\left\{X^k_{ij}\right\}$ and $\left\{C^k_{rs}\right\}$ can be established. Additionally, the denominators of the multiplicative updating equations of both SymNMF and ASymNMF are not well-defined when they approach zeros, which may cause numerical problems.

Proposition 4. *If $A \neq XX^T$, then any stationary point of the optimization problem* (3.17) *is on the boundary of its constraint set $\{X|X \geq 0\}$.*

Proof. By definition, a stationary point of the optimization problem (3.17) should satisfy the Karush-Kuhn-Tucker (KKT) optimality condition [47]:

$$\left((A - XX^T)X\right)_{ij} X_{ij} = 0. \tag{3.58}$$

With the assumption $A \neq XX^T$ in general, we find that the stationary points of (3.17) necessarily contain zero elements ($\exists i, j, X_{ij} = 0$) in X. This implies that the stationary points of (3.17) are on the boundary of $\{X|X \geq 0\}$. □

Proposition 5. *If $A \neq XCX^T$, then a stationary point of the optimization* (3.18) *is on the boundary of its constraint set* $\{(X, C)|X \geq 0, C \geq 0\}$.

Proof. Based on [106], a stationary point of the optimization problem (3.18) should satisfy the following Karush-Kuhn-Tucker (KKT) optimality condition:

$$\begin{cases} \big((XCX^T - A)XC^T + (XC^TX^T - A^T)XC\big)_{ij}X_{ij} = 0; \\ \big(X^T(XCX^T - A)X\big)_{rs}C_{rs} = 0. \end{cases} \tag{3.59}$$

Because in general, $A \neq XCX^T$ and $A^T \neq XC^TX^T$, it requires that there exists $X_{ij} = 0$ or $C_{rs} = 0$ in the stationary point to satisfy the KKT condition, which implies that the stationary points of (3.18) are on the boundary of $\{(X, C)|X \geq 0, C \geq 0\}$.

□

Proposition 6. *If A has neither zero column nor zero row, and the initialization point of SymNMF $X_{ij}^0 > 0, \forall i, j$, then*

$$X_{ij}^k > 0, \forall i, j, \forall k \geq 0. \tag{3.60}$$

Proof. From [47], we know the updating rule of SymNMF is

$$X_{ij}^{k+1} \leftarrow X_{ij}^k\Big(\frac{1}{2} + \frac{(AX^k)_{ij}}{2(X^k(X^k)^TX^k)_{ij}}\Big). \tag{3.61}$$

When $k = 0$, the equation (3.60) holds by the assumption. By induction, if (3.60) is correct at k, then it is correct at $k+1$ too. The nominator and denominator in (3.61) are both strictly positive under the assumption that A has neither zero column nor zero row. Therefore, $X_{ij}^{k+1} > 0$. □

Proposition 7. *If A has neither zero column nor zero row, and the initialization*

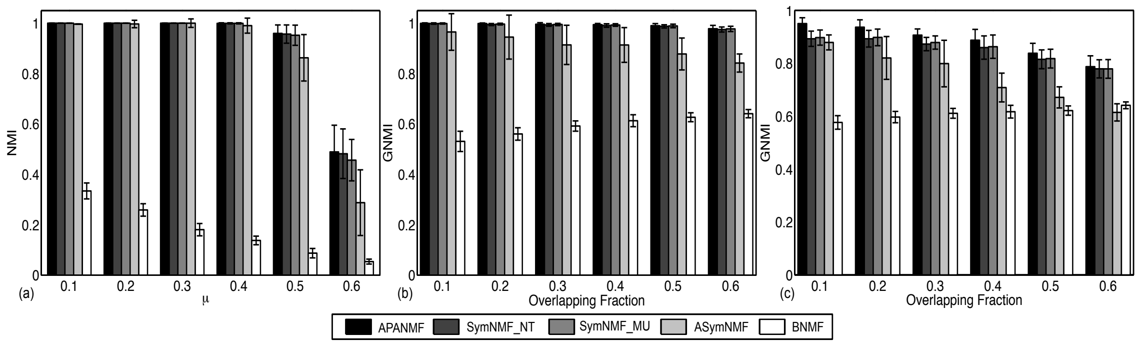


Figure 3.13: Performance comparison for undirected graph clustering: (a) NMI comparison (non-overlapping) with increasing mixing parameter μ. (b) GNMI comparison (overlapping) with increasing overlapping fraction values θ when $\mu = 0.1$. (c) GNMI comparison (overlapping) with increasing overlapping fraction values θ when $\mu = 0.3$.

point of ASymNMF $X_{ij}^0 > 0$ *and* $C_{rs}^0 > 0,\ \forall i, j, r, s,$ *then*

$$X_{ij}^k > 0, C_{rs}^k > 0,\ \forall i, j, r, s,\ \forall k \geq 0. \tag{3.62}$$

Proof. From [106], we know the updating rule of ASymNMF is

$$\begin{aligned}
&X_{ij}^{k+1} \leftarrow X_{ij}^k \cdot \\
&\Big(\frac{(A^T X^k C^k + A X^k (C^k)^T)_{ij}}{\big(X^k(C^k(X^k)^T X^k (C^k)^T + (C^k)^T (X^k)^T X^k C^k)\big)_{ij}}\Big)^{\frac{1}{4}}; \\
&C_{rs}^{k+1} \leftarrow C_{rs}^k \frac{((X^k)^T A X^k)_{rs}}{((X^k)^T X^k C^k (X^k)^T X^k)_{rs}}.
\end{aligned} \tag{3.63}$$

When $k = 0$, the equation (3.62) holds by the assumption. By induction, if (3.62) is correct at k, then it is correct at $k + 1$. Both the nominator and denominator in (3.63) are strictly positive under the assumption that A has neither zero column nor zero row. Therefore, (3.62) holds at $k + 1$, and the proof is complete. □

Through the procedure of APANMF, the dominant computational cost is a relatively cheap matrix multiplication involving the adjacency matrix A. Assuming that PMH and PMB respectively take k and l iterations in average to converge, the time complexity for updating X^t and B^t are $O(kN^2K)$ and $O(lN^2K)$. Furthermore, if APANMF takes t iterations of PMH and PMB steps, then the overall time complexity of APANMF is $O(t(k+l)n^2K)$.

3.3.3.4 Initialization

Our flexible graph clustering formulation is not jointly convex with respect to X and B. Therefore, a good initial point is important to achieve high quality solutions. In this section, we select the initialization points (X^0, B^0) as follows: First, we consider $\Psi(X, B)$ as an unconstrained optimization problem for B with randomly generated X. Setting $\nabla_B \Psi(X, B) = 0$ to obtain

$$\hat{B}^0 = (X^T X)^{-1} X^T A X (X^T X)^{-1}. \tag{3.64}$$

Then for $\Psi(X, B^0)$ we set $\nabla_X \Psi(X, B^0) = 0$ and get

$$\hat{X}^0 = A X \hat{B}^0 (\hat{B}^0 X^T X \hat{B}^0)^{-1}. \tag{3.65}$$

We project $(\hat{X}^0, \hat{B}^0)$ to the non-negative orthant and choose the best (X^0, B^0) that gives the minimum objective function value as our initialization point.

3.3.3.5 Selection of α and β

We explore the stochastic nature of the proposed algorithm to determine α and β. A similar strategy has been adopted in [119]. We propose to estimate the robustness of a specific combination of α and β by measuring the differences and similarities of multiple realizations. For each realization, we compute a connectivity matrix

$C = X^I B^I (X^I)^T$, where X^I and B^I are binary matrices recovered from X and B obtained from the algorithm 3. $X^I_{ij} = 1$ when $X_{ij} \geq \epsilon$ and $X^I_{ij} = 0$ when $X_{ij} < \epsilon$, where ϵ is a user-defined threshold controlling the number of memberships of the overlapping vertices [76]. Similarly, $B^I_{rs} = 1$ if $B_{rs} \geq 0.5$ (meaning the probability of cluster r interacting with cluster s is larger than 0.5), otherwise $B^I_{rs} = 0$. Then we can compute the consensus matrix $\bar{C}$ defined as the average connectivity matrix over many realizations. The entry $\bar{C}_{ij}$ of $\bar{C}$ ranges from 0 to 1 and reveals the probability that vertex i connects to vertex j.

After we obtain $\bar{C}$, we can estimate the entropy, which measures the stability of the common network structure. Assuming $\bar{C}_{ij}$ is independent of each other, we define the entropy score as

$$E_n = \frac{1}{n^2} \sum_{i,j} \left[\bar{C}_{ij} \log(\bar{C}_{ij}) + (1 - \bar{C}_{ij}) \log(1 - \bar{C}_{ij}) \right]. \tag{3.66}$$

For certain α and β, $E_n = 1$ means the network structure is totally unstable ($\bar{C}_{ij} = 0.5$), while $E_n = 0$ indicates that the edges in $\bar{C}$ are perfectly stable ($\bar{C}_{ij} = 1$ or $\bar{C}_{ij} = 0$). We demonstrate that the E_n score can help to select α and β in Section 3.3.4.4.

3.3.4 Experimental results

In this section, in order to show the improved noise tolerance of our new graph clustering formulation and the effectiveness of our novel proximal algorithm APANMF for solving noisy graph clustering, we compare our APANMF with SymNMF_MU [24], SymNMF_NT [47], ASymNMF [106] and BNMF [19] on both synthetic benchmarks under different noise levels as well as real-world large-scale networks.

To demonstrate the robustness of our APANMF with respect to the noise, we explicitly tune the noise level of synthetic networks. Benchmarks for undirected and directed graphs are simulated by the LFR algorithm [49] with different mixing parameters μ to control the noise level. For block modeling benchmarks [111, 109], the Maslov-Sneppen procedure [60] is applied to shuffle different fractions of edges to add in noise. Both the mixing parameter μ and the Maslov-Sneppen procedure have the same effect, which is to perturb the fraction of edges within the correct communities. For simplicity, we use μ to present the noise level for all synthetic networks (undirected and directed benchmarks [49] and block modeling benchmarks [111, 109]). For example, $\mu = 0.1$ means that 10% of correct edges are perturbed to connect to the wrong vertices that do not follow the underlying interaction patterns. Because the perturbation of correct edges simulates the false positive and false negative edges in real-world networks, the robustness of our formulation with respect to potential noise in real-world networks can be verified by testing our APANMF on noisy benchmarks with different μ.

For the same noise level μ, we randomly generate 20 networks. For each random network, we implement each algorithm 10 times and choose the one with the best objective function value as the solution for this network. For all competing algorithms, we stop the algorithms when the objective function value does not decrease more than 0.1. The regularization parameters α and β of APANMF are determined by brute-force search in $S = \{(\alpha, \beta)|\alpha \in \{0, 1, 2, 3, 4, 5\}$ and $\beta \in \{0, 1, 2, 3, 4, 5\}\}$. For every network, we compute the entropy score based on (3.66) for every combination of α and β in S from 10 different realizations (initializations), and we choose the best α and β that yield the minimum entropy score. For λ of BNMF, we use the same procedure and set λ from 0 to 5 with an interval of 1. To quantitatively evaluate the performance of each algorithm for synthetic networks, we use the Normalized Mutual Information (NMI) [5] as the performance index for non-overlapping clustering

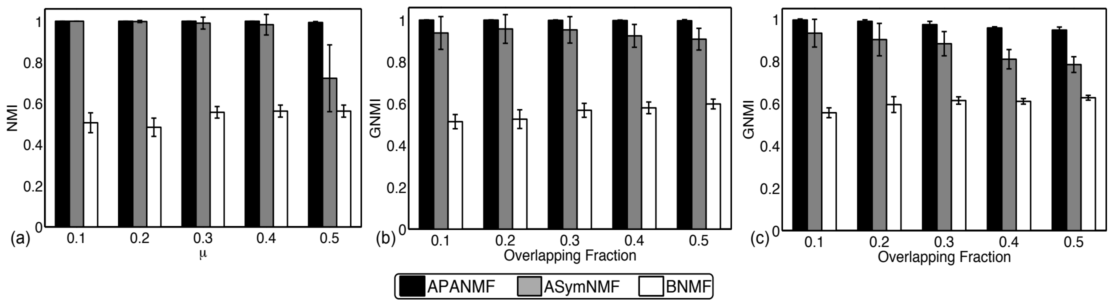


Figure 3.14: Performance comparison for directed graph clustering: (a) NMI comparison (non-overlapping) with increasing mixing parameter μ. (b) GNMI comparison (overlapping) with increasing overlapping fraction values θ when $\mu = 0.1$. (c) GNMI comparison (overlapping) with increasing overlapping fraction values θ when $\mu = 0.3$.

comparison and the Generalized Normalized Mutual Information (GNMI) [50] for overlapping clustering comparison. The evaluation criteria for real-world datasets are introduced in the corresponding sections. All experiments are implemented on a MacBookPro laptop with an Intel i5 dual core processor and 8 GB memory.

3.3.4.1 Undirected graph clustering

To generate undirected graph benchmarks, we adopt the well-known LFR algorithm [49], in which the distributions of vertex degree and cluster size are both based on power laws with tunable exponents. In this section, the benchmark networks are randomly generated based on the similar parameters adopted in [76]: The number of vertices $n = 400$; the average vertex degree is 20 and the cluster size ranges from $c_{min} = 40$ to $c_{max} = 80$. To validate the performance with different parameters, we further tune the mixing parameter (noise level) $\mu = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$, which can be understood as the noise level indicating the portion of a given vertex's edges that connect to the vertices outside the community. This simulates potential noise at different levels in these randomly generated networks. When evaluating the performance for overlapping clustering, we set the overlapping fraction $\theta = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$, which measures the fraction of vertices belonging to more than one clusters.

The comparison among all competing algorithms is shown in Fig. 3.13. The mean values and standard deviations achieved by all competing algorithms for each parameter setting are obtained from 20 randomly generated benchmarks. Fig. 3.13(a) illustrates the performance comparison on non-overlapping benchmarks with various mixing parameters μ. From the figure, we observe that the NMI bar from our APANMF is consistently higher than bars of all the other state-of-the-art algorithms, which indicates that APANMF identifies clusters that are closest to the ground truth. We also notice that our APANMF behaves marginally better than SymNMF_MU and SymNMF_NT, especially for large mixing parameters, which demonstrates that APANMF is more robust to noise than SymNMF_MU and SymNMF_NT since it explicitly enforces the sparsity of B and X.

Fig. 3.13(b) and (c) illustrate the performance for overlapping community detection under $\mu = 0.1$ and $\mu = 0.3$, respectively. With the increasing overlapping fraction values, the difficulty for graph clustering increases. We still find that the GNMI bar of our APANMF is consistently higher than bars of the other competing algorithms with respect to different overlapping fraction values.

Furthermore, we test the statistical significance of our APANMF by comparing APANMF with SymNMF_MU and SymNMF_NT respectively as SymNMF_MU and SymNMF_NT are empirically the best-preforming algorithms in addition to our APNNMF. By two-sample t-test with unequal variances, we find that APANMF performs significantly better than SymNMF_MU and SymNMF_NT at the noise level $\mu = 0.6$ in the experiments illustrated in Fig. 3.13 (a) and (c) at the significant level of 0.05.

3.3.4.2 Directed graph clustering

We further generate directed graph benchmarks by LFR [49]. Similarly, to test the behavior of the competing algorithms, we simulate non-overlapping and overlapping directed benchmarks with different mixing parameters μ (noise levels). We set the number of vertices $n = 400$, the average vertex degree to 20 and the cluster size from $c_{min} = 40$ to $c_{max} = 80$. For non-overlapping directed graph benchmarks, we randomly generate benchmarks with the increasing mixing parameters (noise level): $\mu = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. While for overlapping directed graph benchmarks, we simulate benchmarks with the increasing overlapping fraction values $\theta = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. We compare our APANMF with all the other methods except SymNMF_MU and SymNMF_NT as SymNMF_MU and SymNMF_NT can not handle directed graphs.

Fig. 3.14 shows the comparison results for non-overlapping and overlapping clustering for directed graphs. For non-overlapping clustering comparison shown in Fig. 3.14(a), APANMF and ASymNMF are competitive when the mixing parameter μ is small. However, when it reaches $\mu = 0.5$, APANMF performs significantly better than ASymNMF, which further validates that with high noise level, the sparsity regularization in APANMF can help obtain better results. For overlapping clustering comparison shown in Fig. 3.14(b) and (c), with the increasing overlapping fraction values at fixed $\mu = 0.1$ and $\mu = 0.3$ respectively, the bars of GNMI values obtained by APANMF are consistently higher than other two competing algorithms. Additionally, the GNMI values of APANMF are the most stable one with the smallest standard deviation. Therefore, Figs. 3.14(b) and (c) demonstrate that APANMF is also robust to the overlapping fraction. In summary, obviously our APANMF outperforms ASymNMF and BNMF for both non-overlapping and overlapping clustering

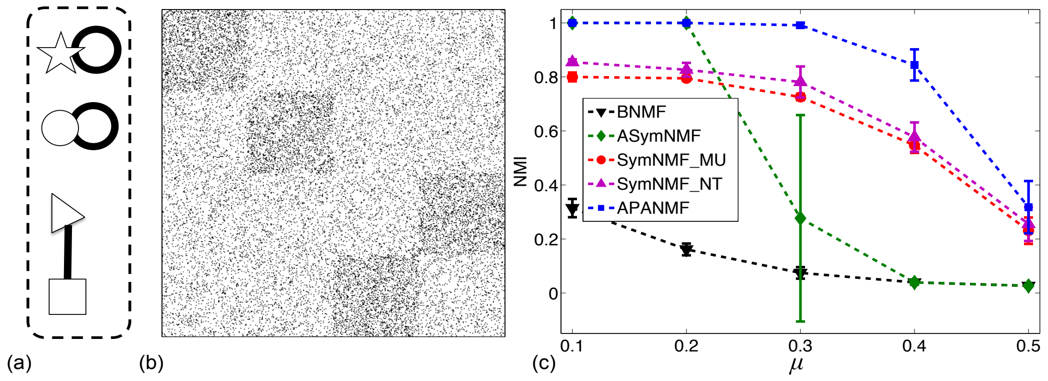


Figure 3.15: (a) Underlying blockmodel structure of synthetic blockmodel benchmarks. (b) Example of a random network with $\mu = 0.4$. (c) NMI comparison with the increasing noise level for all the competing algorithms.

of directed graphs.

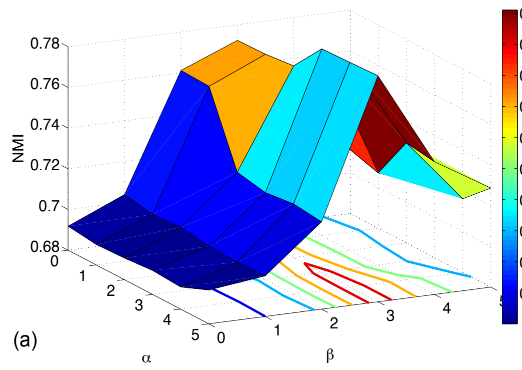
3.3.4.3 Block modeling

Our APANMF can also solve block modeling clustering problems. We generate synthetic networks as similarly done in [111, 109] with known ground truth block structures. The generated benchmark networks have block structures with two densely connected clusters and two clusters with only edges across each other as shown in Fig. 3.15(a). We first simulate the noise-free networks with the size of each block clusters set at 100. To vary the difficulty of the block modeling clustering problem, we instill the noise to the network topology with different levels, which can be controlled by permuting the percentage of correct edges based on the Maslov-Sneppen algorithm [60]. In the Maslov-Sneppen algorithm, two unconnected edges are randomly drawn and then mutually rewired. The noise level μ controls the percentage of correct edges to be permuted. Fig. 3.15(b) provides an example with 40% edges being permuted ($\mu = 0.4$).

Fig. 3.15(c) illustrates the comparison in terms of NMI. From the figure, we observe that the NMI curve of our APANMF is consistently on top of all the other competing algorithms at all noise levels. In addition, we discover that when the noise level is low, both APANMF and ASymNMF have competitive performance, better than the other two algorithms, since the introduction of the image graph B in both methods. However, with the increasing noise level, ASymNMF fails to detect the block structures, which consolidates that our APANMF is more robust to noise with additional sparsity regularization. For SymNMF_MU and SymNMF_NT, as they are designed to identify densely connected clusters, the bipartite-like clusters in Fig. 3.15(a) can not be detected even when there is no noise. Additionally, SymNMF_NT performs marginally better than SymNMF_MU because SymNMF_NT converges to a stationary point. For BNMF, the approximation of M^{ideal} may not capture the latent structure of the graph, which influences its performance.

3.3.4.4 Effect and determination of α and β

The regularization coefficients α and β control the sparsity of X and B in APANMF. The larger α and β are, the sparser X and B become. To discover the relationships among the selection of α and β, clustering accuracy and entropy scores under a high noise level, we implement the following experiment. We randomly generate a synthetic network with the noise level of $\mu = 0.4$. The underlying block structure is the same as illustrated in Fig. 3.15(a). We select the (α, β) pair from $S = \{(\alpha, \beta) | \alpha \in \{0, 1, 2, 3, 4, 5\} \text{ and } \beta \in \{0, 1, 2, 3, 4, 5\}\}$. To demonstrate the effectiveness of the regularization terms, for each initialization we implement our algorithm through all (α, β) pairs in set S. We apply 10 different initializations and compute the average NMI value and entropy score of each (α, β) pair. Fig. 3.16(a) displays the surface of NMI values for every (α, β) pair and Fig. 3.16(b) illustrates the surface of entropy scores for every (α, β) pair. Fig. 3.16(a) shows that better NMI values can be achieved by appropriately selecting (α, β), which further indicates the necessity of using regularization terms to obtain robust results for noisy networks. Additionally, we discover that the best average NMI values and the minimum entropy scores are attained at the same point $(\alpha, \beta) = (5, 3)$, which demonstrates that using entropy scores can help us choose appropriate α and β.



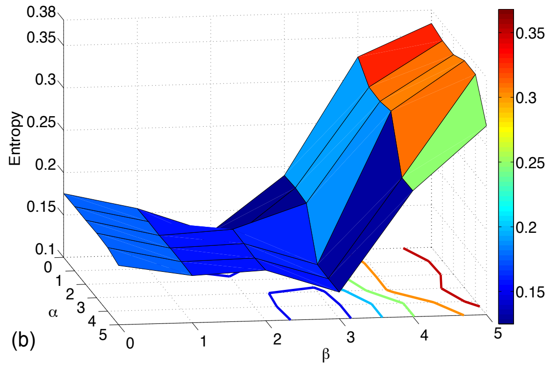


Figure 3.16: (a) Surface of average NMI values for every (α, β) pair. (b) Surface of entropy scores for every (α, β) pair.

3.3.4.5 Facebook ego network

The Facebook ego network is obtained from the SNAP library (http://snap.stanf -ord.edu/data/). The Facebook ego network combines 10 ego networks with 4,039 vertices as Facebook users and 88,234 edges denoting virtual friendship. This combined ego network has manually labeled ground truth from Facebook circles. Our task is to detect the overlapping communities within the ego network. Because we have the ground truth, we can evaluate the performance of all competing algorithms. The measure we applied is the geometric mean of two other measures, which are the cluster-wise sensitivity (Sn) and the cluster-wise positive predictive value (PPV) [53]. Given r predicted and s reference communities, let t_{ij} denote the number of vertices that exist in both predicted community i and reference community

j, and w_j represent the number of vertices in reference community j. Then Sn and PPV can be defined as

$$Sn = \frac{\sum_{j=1}^{s} \underset{i=1,\ldots,r}{max}\, t_{ij}}{\sum_{j=1}^{s} w_j}\ ,\ PPV = \frac{\sum_{i=1}^{r} \underset{j=1,\ldots,s}{max}\, t_{ij}}{\sum_{i=1}^{r}\sum_{j=1}^{s} t_{ij}}. \tag{3.67}$$

We use their geometric mean as our "accuracy" index to balance these two measures ($Acc = \sqrt{Sn \times PPV}$) [53].

We set the number of potential communities $K = 200$ for all the algorithms and choose $\alpha = 5$ and $\beta = 2$ for APANMF and $\lambda = 1$ for BNMF based on the entropy score (3.66). The performance comparison for this Facebook ego network is provided in Table 3.10, from which it is clear that our APANMF obtains the best Sn, PPV, and Acc scores. The results further demonstrate that APANMF is the best graph clustering method for this application. SymNMF_NT fails to implement due to the memory limitation. Hence, SymNMF_NT is not included in Table 3.10.

Table 3.10: Comparison on Facebook ego network.

	APANMF	SymNMF_MU	ASymNMF	BNMF
Sn	**0.4243**	0.3905	0.3978	0.2078
PPV	**0.5731**	0.5614	0.5070	0.2843
Acc	**0.4931**	0.4682	0.4491	0.2431

3.3.4.6 Human protein-protein interaction netwrok

To further illustrate the practical usage of our flexible method in computational biology, we apply all the competing algorithms and compare their performances on a human protein-protein interaction (PPI) network extracted from the PIPs dataset (HsaPIPs) [62]. This HsaPIPs network has 5,445 proteins and 74,686 edges denoting whether two corresponding proteins bind with each other. For this biological network, we do not have the clustering ground truth, which is often the case for most of the real-world network datasets. As typically done in computational biology, we evaluate the performance based on manual curations of genes and/or proteins in this network, for example, based on Gene Ontology (GO) terms [7]. GO terms annotate groups of genes representing certain gene product properties in cells. GO term enrichment analysis [112] can help interpret the corresponding cellular functions for the proteins in detected clusters by statistically detecting whether they correspond to a specific GO term. Assuming a detected cluster has n proteins with m proteins annotated to a GO term and the whole network has n proteins with M proteins annotated with the same GO term. Then the p-value of the identified cluster with respect to the enrichment of proteins within that GO term can be calculated as [112]

$$\text{p-value} = \sum_{i=m}^{n} \frac{\binom{m}{i}\binom{n-M}{n-i}}{\binom{n}{n}}. \tag{3.68}$$

In this implementation, we set $K = 600$ for all the competing algorithms and choose $\alpha = 4$ and $\beta = 2$ for APANMF and $\lambda = 1$ for BNMF based on the entropy score computation (3.66). SymNMF_NT again fails to run due to its memory issue. For performance comparison, a GO term is considered enriched when there is a detected cluster significantly enriched with this GO term with the corresponding p-value less than $1e{-}3$. For each cluster, we choose the lowest p-value of all its enriched GO terms as the corresponding p-value of this cluster. Fig. 3.17 illustrates the performance comparison in terms of the negative logarithms of the p-values for every identified clusters in the descending order. We find that the curve of APANMF is the longest one, which indicates that APANMF detects the largest number of biologically meaningful clusters (420) with the corresponding p-values lower than

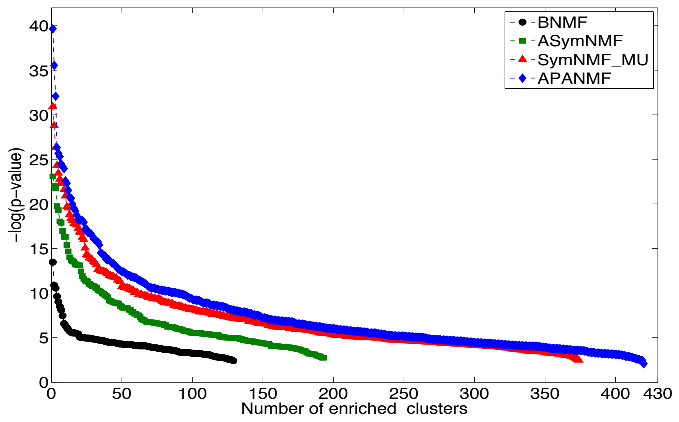


Figure 3.17: GO enrichment comparison for all the competing algorithms.

1e−3. Additionally, we notice that the curve of APANMF is on top of all the other curves, which implies that the significant level of the clusters identified by APANMF is higher than the others. Furthermore, we count the total number of the enriched GO terms obtained by each competing algorithm. We find that 1637 GO terms are enriched by the clusters detected by APANMF. For SymNMF_MU, ASymNMF, and BNMF, 1390, 809, and 283 GO terms are significantly enriched, respectively. Obviously, APANMF covers the largest number of enriched GO terms which indicates that APANMF unearths richer biological information. It is not surprising because many researchers [109, 111] have discovered that PPI networks have block modeling structures and our APANMF is more powerful for discovering the block modeling structures of noisy graphs.

3.3.5 Conclusions

In this section, for clustering noisy networks, we propose a flexible NMF-based formulation with the explicit sparsity regularization of all factorized components. Our new framework is noise tolerant and can solve graph clustering with different settings such as both undirected graph community detection and directed graph clustering with either overlapping or non-overlapping clustering structures, and more general block modeling clustering problems as well. Furthermore, we propose an alternating proximal method APANMF to solve our new optimization problem with the convergence guarantee. The results on synthetic benchmarks and real-world networks demonstrate that our method outperforms other NMF-based state-of-the-art graph clustering algorithms. The proposed APANMF has the potential to derive useful knowledge in diverse applications including social and biological network analysis.

4. BLOCK MODELING FOR MULTIPLE PROTEIN INTERACTION NETWORKS*

In this chapter, we introduce two algorithms for searching for functional modules in pairwise networks. These two algorithms are extend based on the formulations proposed in sections 3.1 and 3.2, respectively.

4.1 Simulated annealing

We formulate network clustering as a block modeling problem by mapping the original networks to an *image graph* in which nodes represent potential functional modules with specific functionalities. The *image graph* optimally preserves the interaction patterns among nodes of the original networks across corresponding modules (Figure 4.1A), and enables to capture the functional interdependences between biomolecules based on the ways they interact with each other. We adopt a simulated annealing (SA) algorithm for the corresponding Potts-model [80] to solve the non-convex problem of simultaneous module identification across two networks, which has been shown to yield high quality results. Our experimental results with both synthetic and real-world PPI networks demonstrate that our new joint clustering algorithm solved by SA (JointSA) outperforms separate block modeling clustering algorithm (SingleSA).

*Part of the content of the first section in this chapter is reprinted with permission from "Joint clustering of protein interaction networks by block modeling" by Yijie Wang and Xiaoning Qian, Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, 1616-1620, 2014, Copyright 2014 by IEEE. Part of the content of the second section in this chapter is reprinted with permission from "Joint clustering of protein interaction networks through Markov random walk" by Yijie Wang and Xiaoning Qian, BMC Systems Biology, 8(Suppl 1): S9, 2014, Copyright 2014 by Bioinformatics.

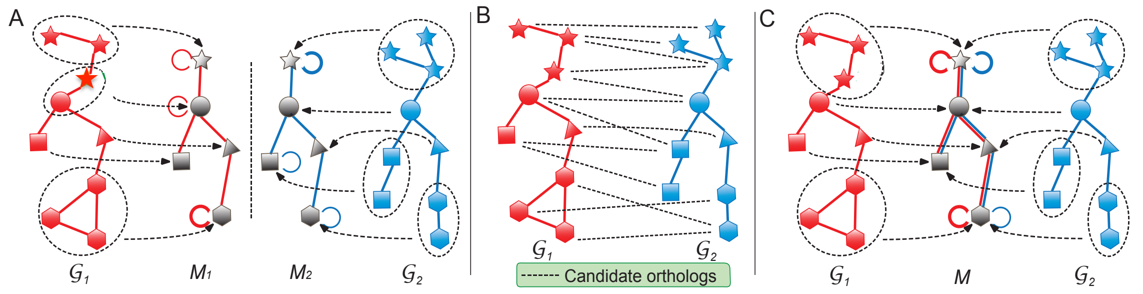


Figure 4.1: A. Network clustering separately for two networks G_1 and G_2; B. Sequence similarities between G_1 and G_2; C. Joint network clustering. The width of edges in the virtual modular space is proportional to the number of aggregated edges in original networks. ©2014 IEEE

4.1.1 Methodology

We first develop an integrated mathematical model for joint clustering of two PPI networks [113]. The motivation is to obtain biologically meaningful results by integrating useful information and accrued knowledge from two networks across species. The essence of the integrated model is to introduce a common virtual network as the image graph for both networks as different species may evolve form the same ancestor and share the same functional modular structure from the perspective of evolution. The virtual network, where each node represents a potential module (either densely connected or sparsely connected) with groups of proteins with similar cellular functions, provides insights into the cellular functional organization by the derived modular structure as shown in Figure 4.1C. We solve the module identification for two networks at the same time by mapping each network into this virtual network. By this integrated model, evolutionarily conserved molecules are more likely to possess independent but coherent functions.

4.1.1.1 Joint network clustering

Let two given PPI networks from two species be $G_k = \{V_k, E_k\}(k = 1, 2)$, where $V_k = \{u_1^k, u_2^k, ..., u_{N_k}^k\}$ are sets of nodes that represent proteins in G_k and E_k are sets of edges representing interactions among proteins in G_k. The network topology of G_k can be represented by adjacency matrix A^k, where $A^k_{ij} \in \{0,1\}$ denotes the interactions between nodes u^k_i and u^k_j in G_k. Let $S_{12}(u^1_i, u^2_j)$ represent the sequence similarity between node $u^1_i \in V_1$ and node $u^2_j \in V_2$. We aim to find mappings from the given networks to the introduced virtual network $M = \{V_M, E_M\}$ as illustrated in Figure 4.1C, where $V_m = \{v^m_1, v^m_2, ..., v^m_{N_m}\}$ denotes the virtual nodes in M and N_m is the total number of virtual nodes in M ($N_m \leq \min_{k\leq 2}\{N_k\}$). The adjacency matrix of M is A^M. For each given network G_k, we define a many-to-one mapping $\Psi_k : V_k \mapsto V_M$ to the virtual network M. For a node u^k_i in G_k, $\Psi_k(i)$ assigns it to a virtual node $v^m_i, 1 \leq i \leq N_m$.

In order to consider the sequence similarities between the nodes across networks and the interaction similarities shared by proteins in two networks, we formulate our joint clustering objective function as follows:

$$\max_{\Psi_1,\Psi_2,A^M} \lambda U(S_{12}, \Psi_1, \Psi_2) + (1-\lambda) Q(A^1, A^2, A^M, \Psi_1, \Psi_2), \tag{4.1}$$

in which λ is a weighting coefficient. Function $U(S_{12}, \Psi_1, \Psi_2)$ computes the total similarity score based on the sequence similarity between corresponding proteins assigned to the same virtual node according to Ψ_1 and Ψ_2 in two networks.

$$U(S_{12}, \Psi_1, \Psi_2) = \sum_{1\leq i\leq N_1}^{1\leq j\leq N_2} S_{12}(u^1_i, u^2_j)\delta_{\Psi_1(i),\Psi_2(j)}, \tag{4.2}$$

where $\delta_{v,v'}$ is the indicator function, which equals to 1 when $v = v'$ and 0 otherwise. Function $Q(A^1, A^2, A^M, \Psi_1, \Psi_2)$ measures the conservation of interaction patterns shared by corresponding proteins assigned to the same module, for which we develop a pairwise block modeling formulation to jointly consider the mapping quality between two networks as well as the interaction patterns within both networks.

Mathematically, for each network G_k, Ψ_k should minimize the mismatch between the given network G_k and the introduced virtual network M [80]:

$$\min_{\Psi_k, A^M} \sum_{i \neq j}^{N_k} \left[A_{ij}^k - A_{\Psi_k(i)\Psi_k(j)}^M\right] (A_{ij}^k - p_{ij}^k), \tag{4.3}$$

in which $\left[A_{ij}^k - A_{\Psi_k(i)\Psi_k(j)}^M\right]$ calculates the number of mismatched edges between G_k and M and $(A_{ij}^k - p_{ij}^k)$ denotes the penalty for the corresponding mismatch. We set $p_{ij}^k = \frac{\sum_{i' \neq i} A_{ii'}^k \sum_{j' \neq j} A_{j'j}^k}{\sum_{i' \neq j'} A_{i'j'}^k}$ to make the total mismatch error on existing edges equal to the error on absent edges $(\sum_{i \neq j}^N A_{ij}^k (A_{ij}^k - p_{ij}^k) = \sum_{i \neq j}^N (1 - A_{ij}^k) p_{ij}^k)$.

Although A^M is unknown before clustering, algebraic manipulations can lead to the absorption of optimizing A^M in the following optimization problem [80]:

$$\max_{\Psi_k} \sum_{m,n}^{N_m} \left| \sum_{i \neq j}^{N_k} (A_{ij}^k - p_{ij}^k) \delta_{\Psi_k(i),m} \delta_{\Psi_k(j),n} \right|. \tag{4.4}$$

Once we derive Ψ_k [80], A^M can be estimated based on the interaction preservation in (4.3) in a straightforward manner. Therefore, for G_1 and G_2, we have $Q(A^1, A^2, A^M, \Psi_1, \Psi_2)$ equal to

$$\sum_{m,n}^{N_m} \left| \sum_{k=1,2} \sum_{i \neq j}^{N_k} (A_{ij}^k - p_{ij}^k) \delta_{\Psi_k(i),m} \delta_{\Psi_k(j),n} \right|. \tag{4.5}$$

To summarize, the final formulation for joint blockmodel clustering can be written as:

$$
\begin{aligned}
&\max_{\Psi_1,\Psi_2} \lambda \sum_{1\le i\le N_1}^{1\le j\le N_2} S_{12}(u_i^1, u_j^2)\delta_{\Psi_1(i),\Psi_2(j)} \\
&+(1-\lambda)\sum_{m,n}^{N_m}\left|\sum_{k=1,2}\sum_{i\neq j}^{N_k}(A_{ij}^k - p_{ij}^k)\delta_{\Psi_k(i),m}\delta_{\Psi_k(j),n}\right|.
\end{aligned}
\tag{4.6}
$$

4.1.1.2 Optimization for joint network clustering

With the new mathematical model for joint network clustering, we now turn to the problem of solving the optimization problem (4.6). To obtain high quality solutions for this highly non-convex problem, we implement a simulated annealing (SA) algorithm based on the heat-bath algorithm for Potts-Models [80].

To derive the SA algorithm, we assume that each potential virtual node v_i^m in M represents a spin state and Ψ_k assigns each original network node to an arbitrary spin state. We use $H_{v_\phi^m}$ to denote the energy of the system represented in the objective function (4.6) with $\Psi_k(i) = v_\phi^m$ at temperature T. We apply the single spin heat-bath update rule, which updates the system energy when making a state change for a given node u_i^k from a state v_ϕ^m to v_α^m: $H_{v_\alpha^m} = H_{v_\phi^m} + \Delta H_{\Psi_k(i):v_\phi^m \to v_\alpha^m}$. The probability of making state assignment change is proportional to the exponential of the corresponding energy change of the entire system with all other nodes' states fixed.

$$
p(\Psi_k(i) = v_\alpha^m) = \frac{\exp\left\{-\beta \Delta H_{\Psi_k(i):v_\phi^m \to v_\alpha^m}\right\}}{\sum_{n=1}^{N_m}\exp\left\{-\beta \Delta H_{\Psi_k(i):v_\phi^m \to v_n^m}\right\}},
\tag{4.7}
$$

in which $\beta = 1/T$. In order to compute the energy change $\Delta H_{\Psi_k(i):v_\phi^m \to v_\alpha^m}$ at T, we first decompose the energy change into two terms based on (4.1):

$$
\Delta H_{\Psi_k(i):v_\phi \to v_\alpha} = \lambda \Delta U_{\Psi_k(i):v_\phi \to v_\alpha} + (1-\lambda)\Delta Q_{\Psi_k(i):v_\phi \to v_\alpha},
\tag{4.8}
$$

where $\Delta U_{\Psi_k(i):v_\phi^m \to v_\alpha^m}$ computes the energy change with respect to node similarities caused by switching u_i^k from the state v_ϕ^m to v_α^m. We can write $\Delta U_{\Psi_k(i):v_\phi^m \to v_\alpha^m}$ as:

$$\Delta U_{\Psi_k(i):v_\phi^m \to v_\alpha^m} = \sum_j^{N_l} S_{kl}(u_i^k, u_j^l) \left[\delta_{\Psi_l(j), v_\alpha^m} - \delta_{\Psi_l(j), v_\phi^m} \right], \tag{4.9}$$

Each potential state change based on (4.9) takes $O(N^{v_\phi^m} + N^{v_\alpha^m})$ operations where $N^{v_\phi^m}$ and $N^{v_\alpha^m}$ denote the number of nodes assigned to states v_ϕ^m and v_α^m. Thus, each local update takes $O((N_1 + N_2)(N^{v_\phi^m} + N^{v_\alpha^m}))$ operations.

The energy change with respect to network structure $\Delta Q_{\Psi_k(i):v_\phi \to v_\alpha}$ can be calculated similarly as in [80] by the following equation

$$\begin{array}{l} \Delta Q_{\Psi_k(i)=v_\phi^m \to v_\alpha^m} = \\ \left(\left| a^k_{v_\phi^m - i, v_\phi^m - i} + a^l_{v_\phi^m, v_\phi^m} \right| - \left| a^k_{v_\phi^m, v_\phi^m} + a^l_{v_\phi^m, v_\phi^m} \right| \right) \\ + \left(\left| a^k_{v_\alpha^m + i, v_\alpha^m + i} + a^l_{v_\alpha^m, v_\alpha^m} \right| - \left| a^k_{v_\alpha^m, v_\alpha^m} + a^l_{v_\alpha^m, v_\alpha^m} \right| \right) \\ +2 \left(\left| a^k_{v_\phi^m - i, v_\alpha^m + i} + a^l_{v_\phi^m, v_\alpha^m} \right| - \left| a^k_{v_\phi^m, v_\alpha^m} + a^l_{v_\phi^m, v_\alpha^m} \right| \right) \\ +2 \sum_{s \neq v_\phi^m, v_\alpha^m}^{N_M} \left(\left| a^k_{v_\phi^m - i, s} + a^l_{v_\phi^m, s} \right| - \left| a^k_{v_\phi^m, s} + a^l_{v_\phi^m, s} \right| \right) \\ +2 \sum_{s \neq v_\phi^m, v_\alpha^m}^{N_M} \left(\left| a^k_{v_\alpha^m + i, s} + a^l_{v_\alpha^m, s} \right| - \left| a^k_{v_\alpha^m, s} + a^l_{v_\alpha^m, s} \right| \right) \end{array}, \tag{4.10}$$

where $a_{r,s}^k$ is the overall mismatch penalty between modules r and s in G_k. $m_{r,s}^k$ represents the total interactions between modules r and s in G_k and D_r^k is the summation of the degrees of all the nodes in module r in G_k. The subscript $v_\phi^m - i$ in (4.10) stands for the operation of removing the corresponding node u_i^k from the set of nodes assigned to v_ϕ^m while $v_\alpha^m + i$ denotes adding the node to v_α^m. These values in (4.10) can be efficiently computed by the following equations:

$$a_{r,s}^k = m_{r,s}^k - \frac{D_r^k D_s^k}{2M^k}; \tag{4.11}$$

$$m^k_{r,s} = \sum_{ij} A^k_{ij} \delta_{\Psi_k(i),r} \delta_{\Psi_k(j),s}; \tag{4.12}$$

$$D^k_r = \sum_{ij} A^k_{ij} \delta_{\Psi_k(i),r}; \quad M^k = \sum_{ij} A^k_{ij}; \tag{4.13}$$

$$a^k_{v^m_\phi - i, v^m_\phi - i} = m^k_{v^m_\phi v^m_\phi} + 2d^k_{i \to v^m_\phi} - \frac{(D^k_{v^m_\phi} - d^k_i)^2}{2M^k}; \tag{4.14}$$

$$a^k_{v^m_\alpha + i, v^m_\alpha + i} = m^k_{v^m_\alpha v^m_\alpha} + 2d^k_{i \to v^m_\alpha} - \frac{(D^k_{v^m_\alpha} + d^k_i)^2}{2M^k}; \tag{4.15}$$

$$a^k_{v^m_\phi - i, v^m_\alpha + i} = m^k_{v^m_\phi v^m_\alpha} - d^k_{i \to v^m_\alpha} + d^k_{i \to v^m_\phi} - \frac{((D^k_{v^m_\phi})^2 - (d^k_i)^2)}{M^k}; \tag{4.16}$$

$$a^k_{v^m_\phi - i, s} = m^k_{v^m_\phi s} - d^k_{i \to v^m_\alpha} - \frac{(D^k_{v^m_\phi} - d^k_i) D^k_s}{2M^k}. \tag{4.17}$$

where $d^k_{i \to v^m_\phi} = \sum_j A^k_{ij} \delta_{j, v^m_\phi}$ denotes the number of interactions between node u^k_i and nodes in state v^m_ϕ and d^k_i denotes the degree of the node u^k_i. Based on the equations (4.10) to (4.17), the local update for ΔQ takes $O((N_1 + N_2) N_m^2)$ operations at each temperature.

4.1.2 Experimental results

To demonstrate that our joint network clustering algorithm (JointSA) is superior to separate network clustering algorithms, we compare our algorithm with the block modeling clustering algorithm of single networks solved by simulated annealing (SingleSA) [80] on synthetic networks and two PPI networks collected from the Database of Interacting Proteins (DIP) [90].

4.1.2.1 Synthetic networks

We test JointSA and SingleSA on a set of synthetic networks. We first generate noise-free networks based on the virtual network shown in Figure 4.2A. In the virtual network, virtual nodes "a", "b" and "e" represent densely connected modules and virtual nodes "c" and "d" represent the modules having the bi-partite structure with interactions running mainly between nodes assigned to them. We set the sizes of the modules corresponding to the virtual nodes "a", "b", "c", "d" and "e" to 16, 48, 32, 32 and 80 respectively. Additionally, we can add noise to networks with the noise level as the percentage of interactions that do not adhere to the topology of these virtual nodes. A similar setting has been used for benchmarking in [72, 80]. For separate network clustering, we apply SingleSA to these randomly generated synthetic networks with different noise levels. For joint network clustering, we simply use two same synthetic networks and further introduce a node similarity S_{12} between them, in which we randomly assign 8 similar pairs in average for each node. We change the difficulty of the joint clustering task by using different noise levels for both network interactions and the nodes similarities. As our JointSA also uses simulated annealing for optimization, we use the same parameters for both SingleSA and JointSA. Both SingleSA and JointSA converge to the final solutions within a few minutes.

As we know the ground truth of the structure in synthetic networks, we use normalized mutual information (NMI) [23] between the ground truth and the clustering results obtained by both algorithms to evaluate the clustering accuracy. Figure 4.2D shows the performance comparison between JointSA and SingleSA. At each noise level, we randomly generate 50 networks. And for each network, we take the best out of 10 runs with different random initializations. The average and the standard

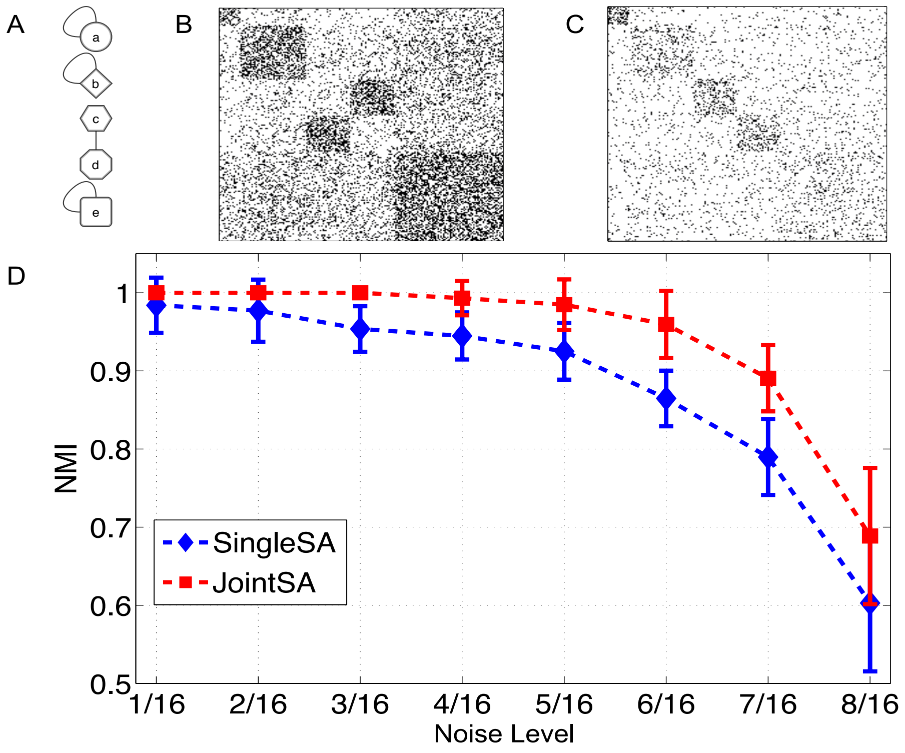


Figure 4.2: Results for synthetic networks with a known underlying structure: A. The structure of the virtual network. B. One example of the adjacency matrix for a generated network at the noise level 0.5; C. One example of the similarity matrix at he noise level 0.5; D. Performance comparison. ©2014 IEEE

deviation of the NMI obtained from 50 randomly generated networks are plotted in Figure 4.2D. Clearly, our JointSA significantly outperforms SingleSA, which implies that the integration of the information across two network including node similarities with reasonable accuracy may significantly improve joint clustering results.

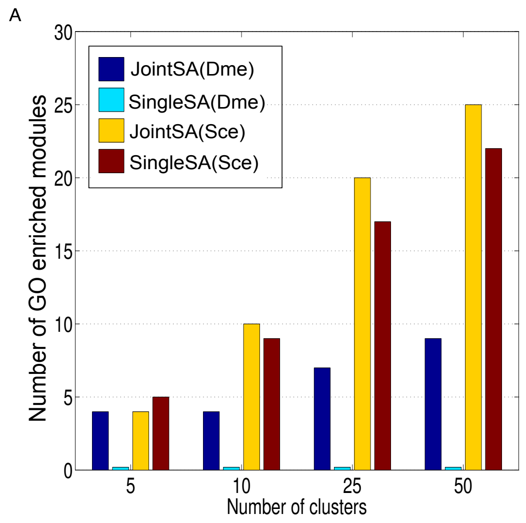
4.1.2.2 Protein interaction networks

In order to validate that joint network clustering can detect more biologically meaningful modules than the separate clustering, we further compare JointSA and SingleSA on real-world PPI networks. We use the PPI networks of *S. cerevisiae* (Sce) and *D. melanogaster* (Dme) extracted from DIP [90]. These two networks have 4,990 nodes with 21,911 edges (Sce) and 7,390 nodes with 22,695 edges (Dme) respectively. The similarities among proteins across two networks are computed by SSEARCH routine in the FASTA package [77]. The final protein similarity is binary by setting to 1 when the e-value between two protein sequences is lower than 10^{-5}, and 0 otherwise. With such large PPI networks, JointSA and SingleSA converge in a few hours.

We annotate each node in PPI networks by its corresponding gene name and use Ontologizer [9] to perform Gene Ontology (GO) enrichment analysis for the results obtained by JointSA and SingleSA. GO enrichment analysis helps interpret the corresponding cellular functions for the proteins in derived modules by statistically detecting whether they correspond to a specific gene ontology category (GO term). Figure 4.3A shows the number of significantly enriched modules detected by both JointSA and SingleSA. From Figure 4.3A, we find that JointSA identifies more GO enriched modules than SingleSA for both *S. cerevisiae* and *D. melanogaster* PPI networks for different number of modules (N_m). Figure 4.3B illustrates the number of enriched GO terms that cover fewer than 100 proteins for the identified modules by both JointSA and SingleSA. We observe that the modules detected by JointSA have more annotated GO terms with smaller sizes (< 100), which implies that the identified modules by JointSA are enriched with more specific cellular functionalities. Hence, JointSA can identify biologically more significant modules with known cell functions.

4.1.3 Discussion

In this section, we propose a novel mathematical framework for joint clustering two PPI networks simultaneously. Furthermore, our mathematical framework (4.1) can be extended to multiple networks in a straightforward manner with the following



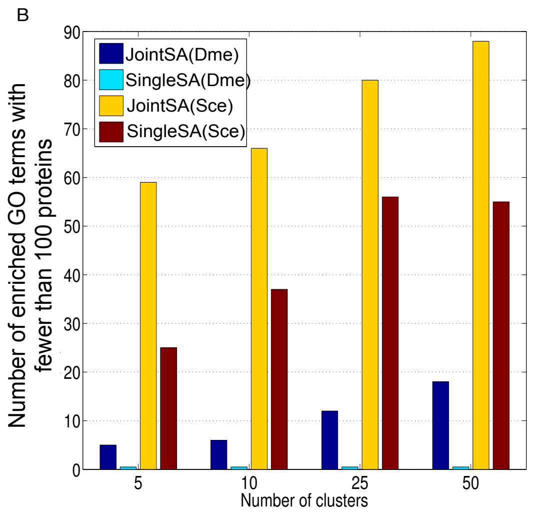


Figure 4.3: A. Number of modules with statistically significantly enriched GO terms below 1% after Bonferroni correction for different N_m. B. Number of statistically significantly enriched GO terms that cover fewer than 100 proteins. ©2014 IEEE

objective function:

$$\begin{aligned} \max_{\Psi_1,...,\Psi_k,A^M} \lambda \sum_{i<j} U(S_{ij}, \Psi_i, \Psi_j) \\ + (1-\lambda) Q(A^1, ..., A^k, A^M, \Psi_1, ..., \Psi_k). \end{aligned} \tag{4.18}$$

We note that the corresponding local update for the SA optimization has $O(\sum_i^k N_i N_m^2)$ computational complexity, which scales linearly with respect to the number of networks k. The convergence of the corresponding simulated annealing solution can be guaranteed by setting the initial temperature high and the cooling down procedure slow [44].

We propose a novel joint clustering formulation which integrates conserved similarity across networks into a flexible clustering model based on block modeling. Our preliminary experimental results have shown that joint clustering outperforms clustering of individual networks separately with respect to obtaining biologically meaningful modules.

4.2 ASModel

There are many existing algorithms for clustering single PPI networks. Normalized cut (NCut) method [95] aims to partition the network based on a novel global criterion, which focuses on the contrast between the total dissimilarity across different clusters and the total similarity within clusters based on network topology. The formulation of NCut is equivalent to finding low conductance sets on the transition matrix of the Markov random walk on the network to analyze [111, 115]. Markov CLustering algorithm (MCL) [26] detects clusters based on stochastic flow simulation, which has been proven to be effective at clustering biological networks. Recently, an enhanced version of MCL—Regularized MCL (RMCL) [91, 93]—has been proposed to penalize large clusters at each iteration of MCL to obtain more balanced clusters and it has been shown to have better performance to identify clusters with potential functional specificity.

However, it is well known that the current public PPI datasets are quite noisy and there exist both false positive and false negative interactions due to different technical reasons [25]. Therefore, clustering simply based on one network constructed from a single data source may not be able to yield robust and accurate results. We may need to appropriately integrate multiple information sources to repress the noise in existing PPI datasets by borrowing strengths from each other. AlignNemo [22] is one of such recent efforts, which detects network clusters on an alignment network of two given PPI networks. AlignNemo takes into account not only the network topology from two PPI networks but also the homology information between proteins across two networks. However, based on the reported experiments and our empirical findings, AlignNemo has low clustering coverage because the alignment network is constructed based on only similar proteins by their sequence similarity and those proteins that do not appear in the alignment network are never considered for clustering.

In this section, we propose a novel joint clustering algorithm based on a new Markov random walk on an integrated network, which is constructed by integrating protein-protein interactions in given PPI networks as well as homological interactions introduced by sequence similarity between proteins across networks. A novel alternative random walk strategy is proposed on the integrated network with the transition matrix integrating both topology and homology information. We formulate the joint clustering problem as searching for low conductance sets defined by this transition matrix. We then derive an approximate spectral solution algorithm for joint network clustering.

4.2.1 Methodology

4.2.1.1 Terminology

Let $G = (U, D)$ and $\mathcal{H} = (V, E)$ be two PPI networks, where U and V are node sets representing N_1 and N_2 proteins in two networks, respectively; and D and E denote edges corresponding to respective protein-protein interactions. We assume that G and $\mathcal{H}$ are connected networks, whose topology structures can be mathematically captured by their corresponding adjacency matrices A_1 and A_2:

$$A_1(i,j) = \begin{cases} 1 & (u_i, u_j) \in D,\, i \neq j; \\ 0 & otherwise. \end{cases} \qquad A_2(i,j) = \begin{cases} 1 & (v_i, v_j) \in E,\, i \neq j; \\ 0 & otherwise. \end{cases} \tag{4.19}$$

where $u_i, u_j \in U$ and $v_i, v_j \in V$ and we first ignore self-loops in PPI networks. Suppose some of the proteins in U and V are known *a priori* to be similar to each other by some criteria, such as their constituent or functional similarity. For example, we compute protein sequence similarity based on the normalized BLAST bit score [3] in this section so that the latter performance evaluation in our experiments based on curated functional annotations is as unbiased as possible. In a similarity matrix S_{12}, each element $S_{12}(u_i, v_j)$ records the similarity between proteins $u_i \in U$ and $v_j \in V$:

$$S_{12}(u_i, v_j) = \frac{\mathrm{BLAST}(u_i, v_j)}{\sqrt{\mathrm{BLAST}(u_i, u_j)} \times \sqrt{\mathrm{BLAST}(v_j, v_j)}} \tag{4.20}$$

where $\mathrm{BLAST}(u_i, v_j)$ stands for the bit score of sequence similarity between proteins u_i and v_j by BLAST [3]. Based on (4.20), we note that $S_{12}(u_i, v_j)$ is in the range $[0, 1]$.

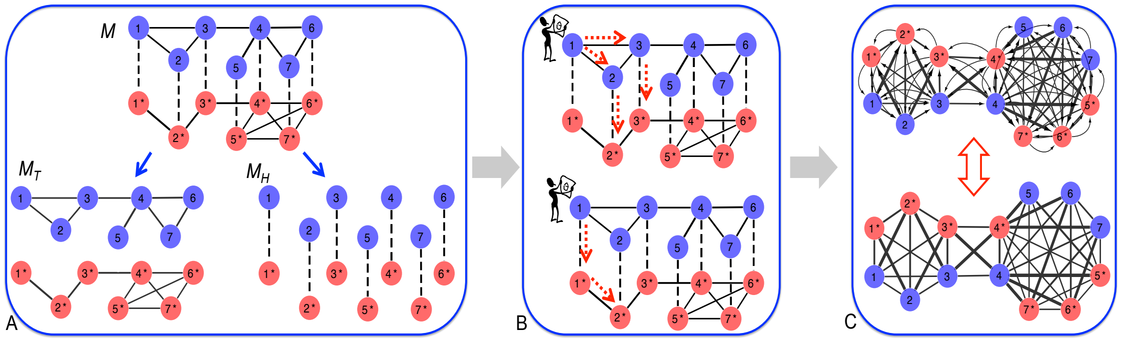


Figure 4.4: Illustration of our proposed joint clustering algorithm. A. Construction of the integrated network. B. Random walk strategy. C. Equivalence between a directed network (transition matrix P) and a symmetric undirected network (transition matrix $\bar{P}$).

4.2.1.2 Integrated network

In order to jointly cluster two PPI networks, we first define a new integrated network $\mathcal{M} = (\mathcal{W}, E_T, E_H)$. The set of nodes $\mathcal{W}$ in this integrated network is the union of proteins in two PPI networks ($\mathcal{W} = U \cup V$). The integrated network $\mathcal{M}$ has two types of interactions, where E_T represents the union of the sets of protein-protein interactions within the PPI networks ($E_T = D \cup E$) and E_H are new “interactions” across two PPI networks introduced by the homological similarity S_{12}. One example of an integrated network is illustrated in Fig 4.4A. In this example, $\mathcal{W}$ contains all the nodes in blue and red colors from two respective networks. The solid edges indicate the interactions in E_T and the dashed edges represent the interactions in E_H.

The integrated network combines both the topology information within two PPI networks and the homology information across two PPI networks. Therefore, $\mathcal{M}$ can be considered as the integration of two networks $\mathcal{M}_T = (\mathcal{W}, E_T)$ and $\mathcal{M}_H = (\mathcal{W}, E_H)$, which share the same set of nodes $\mathcal{W}$. $\mathcal{M}_T$ is the network carrying the topology information within two PPI networks, whose adjacency matrix can be represented as follows:

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}_{N \times N} \tag{4.21}$$

where $N = N_1 + N_2$. $\mathcal{M}_H$ is the network containing the homology information across two networks, whose adjacency matrix can be represented as

$$S = \begin{bmatrix} 0 & S_{12} \\ S_{12}^T & 0 \end{bmatrix}_{N \times N} . \tag{4.22}$$

The examples of $\mathcal{M}_T$ and $\mathcal{M}_H$ are also illustrated in Fig. 4.4A.

4.2.1.3 Random walk strategy on the integrated network

As shown in the previous section, the integrated network contains both topology and homology information represented in two sets of edges. In order to bring strengths from each other to improve the clustering performance in individual networks, we propose a random walk strategy on the integrated network $\mathcal{M}$ to integrate all information sources. To make use of both topology and homology information, we require the random walker must walk through topological and homological interactions (E_T and E_H) in an alternative order. However, as shown in Fig. 4.4B, the random walker can either first walk by $\mathcal{M}_T$ then on the network $\mathcal{M}_H$ or first walk on $\mathcal{M}_H$ then on $\mathcal{M}_T$. For the first type of random walk illustrated in Fig.4.4B, the transition matrix $P_{A\bar{S}}$ can be calculated as

$$P_{A\bar{S}} = P_A \times P_{\bar{S}} \tag{4.23}$$

where $P_A = D_A^{-1}A$ and $P_{\bar{S}} = D_{\bar{S}}^{-1}\bar{S}$. The matrix D_A is a diagonal matrix with the degree of each node on its diagonal elements. $\bar{S} = S + I_{N\times N}$ is the adjacency matrix of network $\mathcal{M}_H$ with self-loops indicating self similarity of proteins. $D_{\bar{S}}$ is the corresponding diagonal matrix with $D_{\bar{S}}(i,i) = \sum_j \bar{S}(i,j)$, where $i,j \in \{1,2,...,N\}$ are new node indices in the integrated network and $\bar{S}(i,j) > 0$ when i,j indicate proteins from different PPI networks. Again, $\bar{S}(i,i) = 1$ for self similarity. Furthermore, we find that P_A is the transition matrix of the random walk on $\mathcal{M}_T$ and $P_{\bar{S}}$ is the transition matrix of the random walk on $\mathcal{M}_H$ including self-loops.

For the second type of random walk illustrated in Fig. 4.4B, we can similarly compute the transition matrix

$$P_{S\bar{A}} = P_S \times P_{\bar{A}} \tag{4.24}$$

where $P_S = D_S^{-1}S$ and $P_{\bar{A}} = D_{\bar{A}}^{-1}\bar{A}$. Here, D_S is a diagonal matrix with $D_S(i,i) = \sum_j S(i,j)$. Here, $\bar{A}$ is the adjacency matrix of $\mathcal{M}_T$ with self-loops to allow for the possibility of random walker staying at the current node. $D_{\bar{A}}$ is the corresponding diagonal matrix with the node degree in $\bar{A}$ on its diagonal. P_S is the transition matrix of the random walk on $\mathcal{M}_H$ and $P_{\bar{A}}$ is the transition matrix of the random walk on $\mathcal{M}_T$ including self-loops.

We further assume that the probability of taking the first type of random walk should be the same as going with the second type of random walk. Therefore, our final transition matrix for the new random walk strategy can be represented by

$$P = \frac{1}{2} P_{A\bar{S}} + \frac{1}{2} P_{S\bar{A}} \tag{4.25}$$

4.2.1.4 Searching for low conductance sets based on P

In $\mathcal{M}_T$, proteins with topological interactions E_T are likely to participate in similar cellular functions. Also, proteins with larger homological interactions E_H in $\mathcal{M}_H$ are more probable to be functionally similar. Because the random walk on the integrated network considers both types of interactions, each element $P(i,j)$ of the corresponding transition matrix can be understood as the probability that proteins i and j have similar functions as these proteins are more likely to reach each other with a larger $P(i,j)$. Based on this, we can make use of the concept of the conductance defined on the Markov chain to identify clusters based on P [95, 92] by searching for low conductance sets.

Similarly as done in [95, 92], we can formulate the optimization problem for joint network clustering:

$$\min \sum_{h=1}^{k} \Phi_P(C_h, \bar{C_h}) \quad s.t. \bigcup_{h=1}^{k} C_h = \mathcal{W}; C_h \cap C_l = \emptyset, \forall h \neq l. \tag{4.26}$$

where $\Phi_P(C_h, \bar{C_h})$ is the defined conductance of node subset C_h to the rest of the network $\bar{C_h}$; and k is the number of desired subsets as final network clusters. The conductance $\Phi_P(C_h, \bar{C_h})$ can be computed as

$$\Phi_P(C_h, \bar{C_h}) = \frac{\sum_{i \in C_h, j \in \bar{C_h}} \pi_i P(i,j)}{\sum_{i \in C_h} \pi_i}, C_h \cup \bar{C_h} = \mathcal{W}, \tag{4.27}$$

where π is the stationary distribution of the corresponding Markov random walk on the integrated network and $P^T\pi = \pi$.

The goal now is to find k low conductance sets defined by P. As in [92], we find that if we consider P as the transition matrix for a directed graph and try to find k low conductance sets based on (4.26), it is in fact equivalent to find k low conductance sets on an undirected graph with another transition matrix $\bar{P}$:

$$\bar{P} = \frac{\pi P + P^T \pi}{2}. \tag{4.28}$$

Due to the equivalence, our optimization formulation for finding k low conductance sets can be formulated finally as

$$\begin{array}{ll} \max & trace\left(\frac{X^T \bar{P} X}{X^T D_{\bar{P}} X}\right) \\ s.t. & X1_k = 1_N, x_{i\ell} \in \{0,1\}, \end{array} \tag{4.29}$$

where $D_{\bar{P}}$ is a diagonal matrix with $D_{\bar{P}}(i,i) = \sum_j \bar{P}(i,j)$; X is a $N \times k$ assignment matrix whose element $x_{i\ell}$ denotes whether node i belongs to cluster ℓ; 1_k and 1_N are all one vectors with k and N elements, respectively. Here, equations (4.26) and (4.29) have been proven to be equivalent previously in [92]. We can derive a spectral method to solve the above problem based on [111]. The directed network with P and its equivalent undirected network with $\bar{P}$ are illustrated in Fig. 4.4C.

4.2.2 Joint clustering algorithm (ASModel)

Our joint clustering algorithm can be summarized into three steps which are illustrated in Fig. 4.4. The first step is to construct the integrated network $\mathcal{M}$. The second step is to compute the transition matrix P based on the alternative random walk strategy in (4.25). The final step is to find low conductance sets on the equivalent network and apply the spectral method to solve the optimization problem. Algorithm 1 provides the pseudo code for ASModel.

Algorithm 1. ASModel for Joint Network Clustering

```
Input: Adjacency matrices $A_1$ and $A_2$, Sequence similarity matrix $S_{12}$, and the number of desired clusters $k$
Output: Cluster assignment matrix $X$
1. Construct the integrated network $\mathcal{M}$ and compute $A$ and $S$;
2. Compute the transition matrix $P$ based on the random walk strategy using (4.25);
3. Obtain the equivalent adjacency matrix $\bar{P}$ which has the same low conductance sets as $P$;
4. Using the spectral algorithm to find $k$ low conductance sets by $\bar{P}$ from (4.29) [111].
```

4.2.3 Experiments

4.2.3.1 Algorithms, data, and metrics

We compare our joint clustering algorithm ASModel to NCut [95], MCL [26], RMCL [91, 93], and AlignNemo [22]. Among the selected algorithms for performance comparison, AlignNemo [22] is a recently proposed protein complex detection algorithm, which also takes into account the homology and topology information from two PPI networks. NCut is equivalent to searching for low conductance sets by the transition matrix defined directly based on the given single network. Therefore, comparing with NCut aims to show that finding low conductance sets on the integrated network by our new ASModel is superior to separately finding similar low conductance sets on individual networks. MCL and RMCL are two state-of-the-art algorithms which have been proven effective on analyzing biological networks. Comparing with them can further demonstrate that our joint clustering algorithm ASModel can achieve better performances than clustering single networks separately. Both NCut and ASModel have one input parameter, which is the number of clusters k. We sample k in $[100, 3000]$ with an interval of 100 and report the best results.

MCL also has one parameter, the inflation number. We similarly search for the best performing value from 1.2 to 5.0 with an interval of 0.1. For RMCL, we adopt the parameters suggested in [91, 93].

We evaluate the performances of ASModel, NCut, MCL, RMCL, and AlignNemo on public PPI datasets for *S. cerevisiae* (budding yeast) and *H. sapiens* (human). For *S. cerevisiae*, *Sce*DIP and *Sce*BGS are two extracted PPI networks from the Database of Interacting Proteins (DIP) [90] and BioGRID [17], respectively. For *H. sapiens*, *Hsa*HPRD and *Hsa*PIPs are corresponding PPI networks derived from Human Protein Reference Database (HPRD) [82] and the PIPs dataset [62]. The details of each PPI network are given in Table 4.1.

In order to access the performance of the competing algorithms, we first implement complex prediction to assess the quality of clustering results by evaluating the agreement of the clusters found by each method with curated protein complex standards. SGD [37] and CORUM [87] complexes are considered as the golden standards for complex prediction for yeast and human PPI networks, respectively. We then implement the GO enrichment analysis for further validation on function predicting performance from clustering results. In order to focus on more specific cellular functions, we only take specific GO terms whose information content (IC) is larger or equal to 2. The information content of a GO term g is defined as:

$$IC(g) = -log(|g|/|root|), \tag{4.30}$$

where $|g|$ and $|root|$ are the number of proteins in GO term g and the number of proteins in its corresponding GO category (biological process, molecular function or cellular component). The information of reference complex datasets and GO terms is also provided in Table 4.1.

We adopt the widely used F-measure [96] to evaluate the performance for complex prediction. F-measure is the harmonic mean of precision and recall: $F = 2 \times$ precision $\times$ recall/(precision + recall), where precision and recall are defined as follows:

$$\text{precision} = \frac{|\{C_i \in C | NA(C_i, R_j) > 0.25, \exists R_j \in R\}|}{|C|}; \tag{4.31}$$

$$\text{recall} = \frac{|\{R_i \in R | NA(C_i, R_j) > 0.25, \exists C_i \in C\}|}{|R|}, \tag{4.32}$$

where $C = \{C_1, C_2, ..., C_k\}$ are the identified clusters by different algorithms and $R = \{R_1, R_2, ..., R_l\}$ denote the corresponding reference complex sets. The neighbor affinity $NA(C_i, R_j) = \frac{|C_i \cap R_j|^2}{|C_i| \times |R_j|}$ measures the overlap between the predicted complex C_i and the reference complex R_j.

To evaluate the performance of GO enrichment analysis, we compute the p-value and the number of enriched GO terms from clustering results. Suppose that the whole network has N proteins with M proteins annotated with one GO term and the detected cluster has n proteins with m proteins annotated with the same GO term. The p-value of the cluster with respect to that GO term can be calculated as [97]

$$\text{p-value} = \sum_{i=m}^{n} \frac{\binom{m}{i}\binom{N-M}{N-i}}{\binom{N}{n}}. \tag{4.33}$$

We choose the lowest p-value of all enriched GO terms in the derived cluster as its final p-value. A GO term is enriched when the p-value of any cluster corresponding to this GO term is less than $1e{-}3$.

Table 4.1: Information of four real-world PPI networks.

Network	#. nodes	#. edges	SGD	CORUM	$\|GO\|$
*Sce*DIP	4980	22076	305	—	956
*Sce*BGS	5640	59748	306	—	1005
*Hsa*HPRD	9269	36917	—	1294	4755
*Hsa*PIPs	5226	37024	—	1193	4560

Table 4.2: The information of the derived clusters by all competing algorithms

PPI	Method	NCut	MCL	RMCL	ASModel(DB)	ASModel(HP)	ASModel(DH)
*Sce*DIP	#. clusters	525	659	814	737	—	702
	coverage	2572	3630	3725	**4537**	—	4425
*Sce*BGS	#. clusters	414	338	772	704	—	—
	coverage	4879	3544	**5210**	5169	—	—
*Hsa*HPRD	#. clusters	981	1239	1508	—	1113	1231
	coverage	6534	7800	6879	—	8631	**8729**
*Hsa*PIPs	#. clusters	491	576	581	—	560	—
	coverage	**4542**	4134	3966	—	4358	—

DB is short for DIP+BGS. HP is short for HPRD+PIPs. DH is short for DIP+HPRD.

4.2.3.2 Joint clustering of PPI networks within the same species

In this section, we first jointly cluster two PPI networks from the same species to demonstrate the effectiveness of our ASModel. Through applying ASModel, we expect that each PPI network can borrow strengths from the other PPI network to enhance the clustering performance.

4.2.3.3 Joint clustering of the SceDIP and SceBGS PPI networks

Complex Prediction

For the *Sce*DIP and *Sce*BGS networks, we report the performance of ASModel, NCut, MCL, RMCL, and AlignNemo on complex prediction in terms of the number of matched reference complexes and F-measure. The detailed information such as the number of clusters (cluster size ≥ 2) and the coverage is listed in Table 4.2. Figs. 4.5A and 4.5B show the comparison results for the number of matched reference complexes

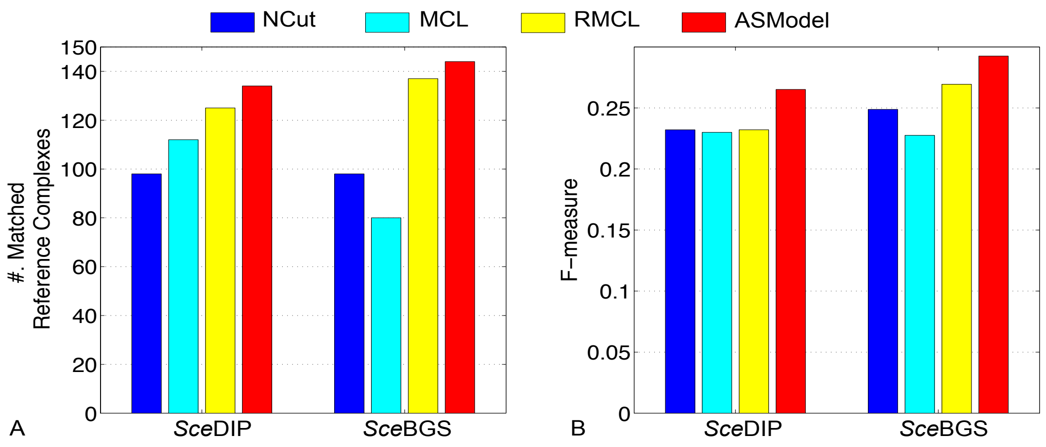
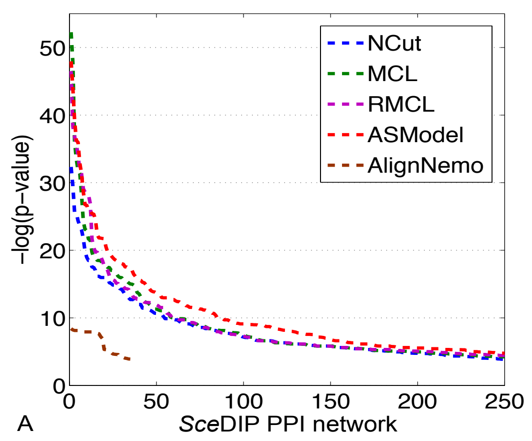


Figure 4.5: Performance comparison of competing algorithms for complex prediction in both yeast PPI networks. A. Comparison on the number of matched reference complexes. B. Comparison on the F-measure.

and F-measure, respectively. As illustrated in Fig. 4.5, ASModel detects the largest number of matched reference complexes and achieves the highest F-measure for both networks, which is substantially better than the results obtained by individual clustering using all the other single network clustering algorithms. Although AlignNemo also uses both topology and homology information, it is interesting to observe that it does not detect any matched reference complexes in this set of experiments, which in fact is different from the reported results in [22] though different networks were analyzed. This may indicate that the random walk strategy in our ASModel better integrates available information across networks than the heuristic strategy adopted in AlignNemo. Another important reason could be that the authors of AlignNemo in [22] adopted different criteria to consider matched complexes while we here adopt more strict evaluation metrics.

GO Enrichment Analysis

GO enrichment analysis has been done based on the detected clusters by ASModel, NCut, MCL, RMCL, and AlignNemo. For each cluster, it may be enriched



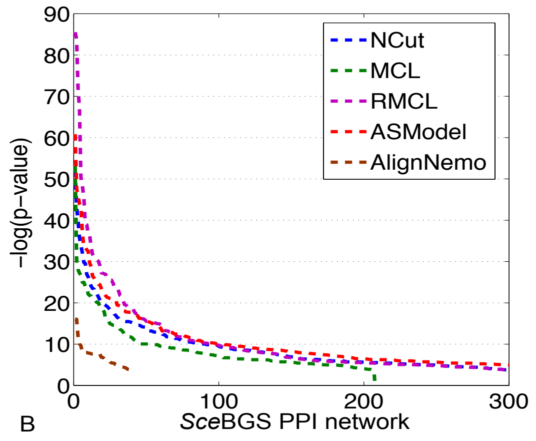


Figure 4.6: Performance comparison of competing algorithms for GO enrichment analysis. A. GO enrichment comparison on the *Sce*DIP network. B. GO enrichment comparison on the *Sce*BGS network.

in multiple GO terms and we choose the lowest p-value as the p-value for the cluster as explained earlier. We first sort the p-values of all clusters in an ascending order and then draw the corresponding monotonically decreasing $-log$(p-value) curves for all the algorithms in Fig. 4.6. As shown in Fig. 4.6A, for the *Sce*DIP PPI network, the curve of ASModel is on top of all the other competing algorithms, which indicates that the clusters detected by joint clustering ASModel are more consistent to the curated GO terms and hence capture the cellular functionalities better. For the *Sce*BGS PPI network from Fig. 4.6B, we find that the curve of RMCL is on top of other algorithms for around the top 80 most significantly enriched clusters. However, when we check more derived clusters, the curve of ASModel is again on top of the other algorithms. Hence, overall, especially when we consider the total number of enriched GO terms shown in Fig. 4.7, functional consistency of the detected clusters is improved by our joint clustering algorithm ASModel as ASModel can identify more enriched GO terms to unearth more biologically meaningful clusters with more significant p-values overall.

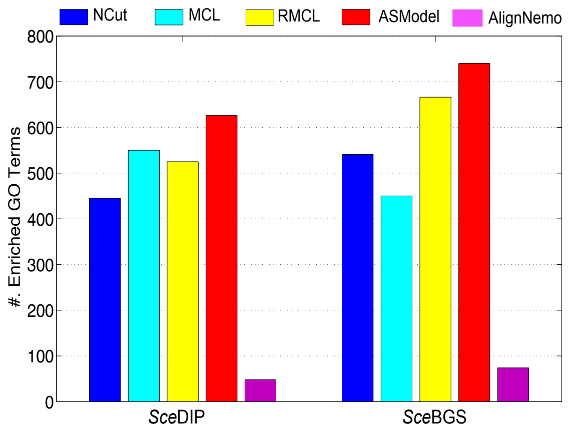


Figure 4.7: Comparison on the number of enriched GO terms for all the competing algorithms in two yeast networks.

In summary, from both complex prediction and GO enrichment analysis, ASModel can achieve more biologically meaningful results. These promising results imply that joint clustering can improve the clustering performance for every individual PPI network when we integrate information from them appropriately.

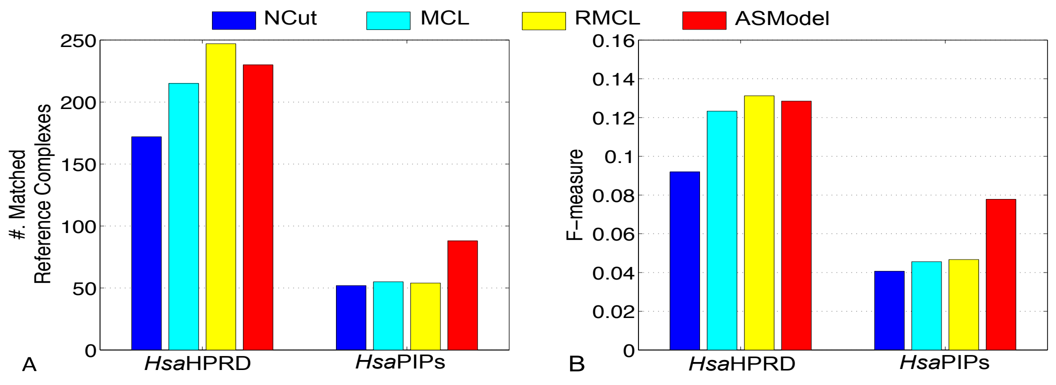


Figure 4.8: Performance comparison of competing algorithms for complex prediction in both human PPI networks. A. Comparison on the number of matched reference complexes. B. Comparison on the F-measure.

Complex Prediction

Similarly, the results of complex prediction from all the competing algorithms on two human PPI networks are shown in Fig. 4.8. For the *Hsa*HPRD network, we find that RMCL and ASModel detect competitive numbers of reference complexes and achieve competitive F-measures. When we check the *Hsa*PIPs network, Fig. 4.8 shows that ASModel identifies much more matched reference complexes and obtain substantially better F-measure than all the other algorithms. AlignNemo again does not detect any matched reference complexes based on the neighbor affinity metric. The performance of ASModel demonstrates that the clustering of *Hsa*PIPs network does benefit from the information in the *Hsa*HPRD network to achieve the better complex prediction performance. However, the performance on the *Hsa*HPRD network is not influenced much, probably due to the incompleteness of the *Hsa*PIPs dataset.

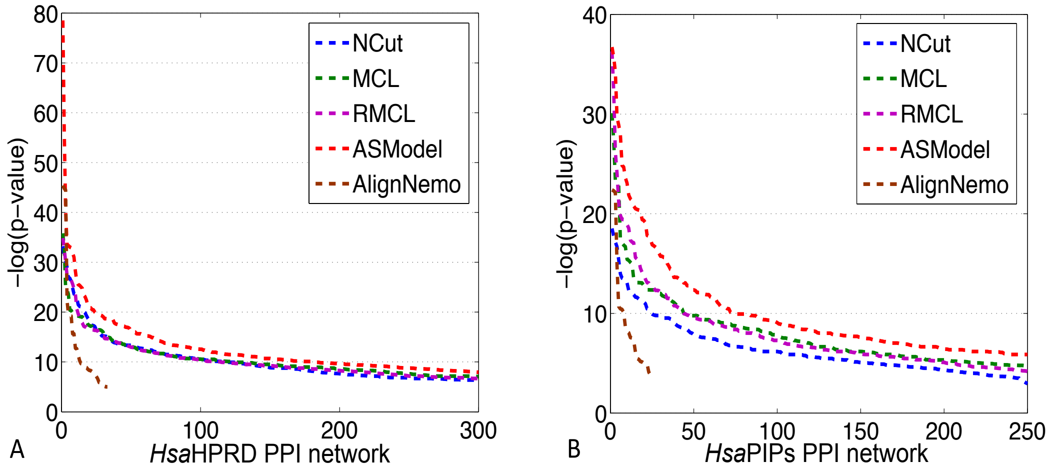


Figure 4.9: Performance comparison of competing algorithms for GO enrichment analysis. A. GO enrichment comparison on the *Hsa*HPRD network. B. GO enrichment comparison on the *Hsa*PIPs network.

GO Enrichment Analysis

We compare ASModel to NCut, MCL, RMCL, and AlignNemo on GO enrichment analysis by drawing similar $-log$(p-value) curves of the top ranked clusters based on their enrichment significance. From Fig. 4.9, we observe that for both human PPI networks, the curves of ASModel are on top of all the competing algorithms. Furthermore, as shown in Fig. 4.10, we find that ASModel also detects the largest number of enriched GO terms on both networks. The overall performance of GO enrichment analysis further validates that joint clustering significantly enhances the clustering performance for each PPI network.

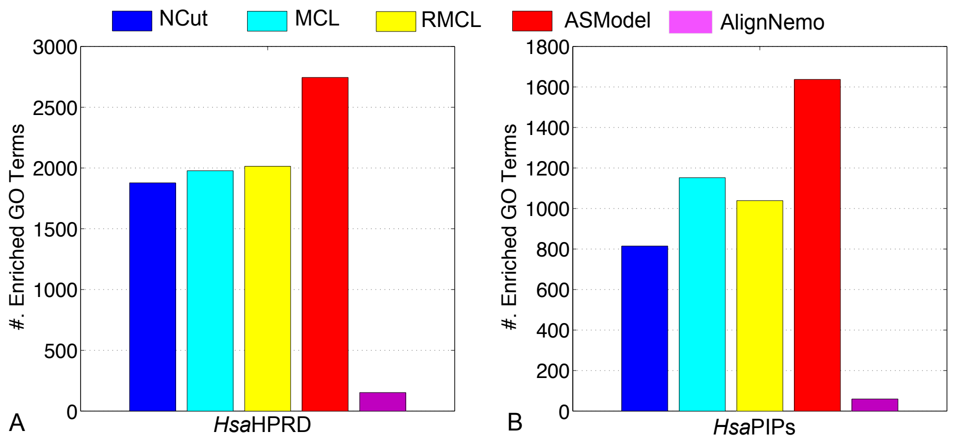


Figure 4.10: Comparison on the number of enriched GO terms for all the competing algorithms in two yeast networks.

From these two experiments of joint clustering PPI networks from the same species, we note that ASModel can make full use of topology and homology information to improve the clustering performance for each PPI network.

4.2.3.5 Joint clustering of PPI networks from different species

Joint clustering of PPI networks within the same species has been proven to yield promising results. In order to show that ASModel can also improve the clustering performance for PPI networks from different species, we have done the following experiment.

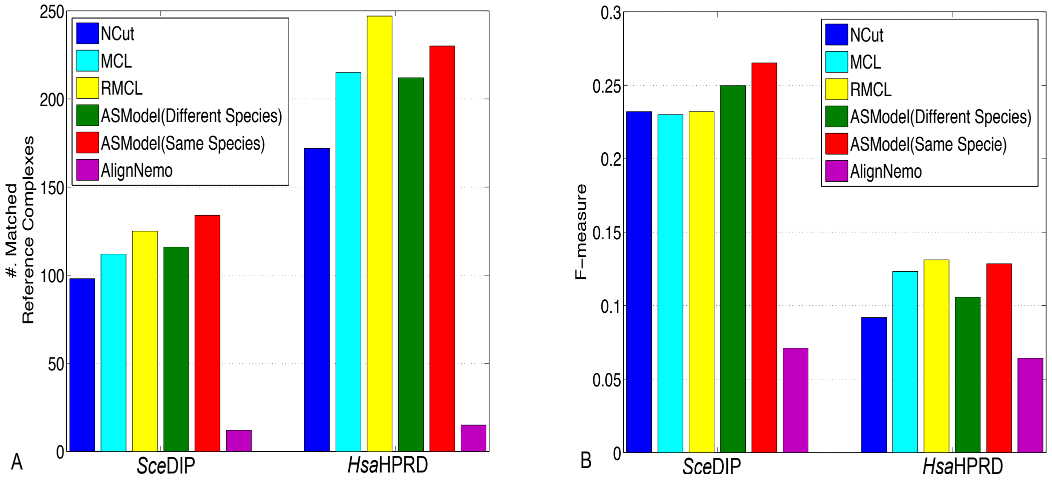


Figure 4.11: Performance comparison of competing algorithms for complex prediction in the *Sce*DIP and *Hsa*HPRD network. A. Comparison on the number of matched reference complexes. B. Comparison on the F-measure. ASModel (Different Species) indicates the results obtained by joint clustering of the *Sce*DIP and *Hsa*HPRD PPI networks. ASModel (Same Species) indicates the results obtained from joint clustering of the *Sce*DIP and *Sce*BGS networks for yeast and joint clustering of the *Hsa*HPRD and *Hsa*PIPs PPI networks for human, respectively.

4.2.3.6 Joint clustering with SceDIP and HsaHPRD PPI networks

Complex Prediction

We first report the performance for protein complex prediction. For the *Sce*DIP network, we compare the results of joint clustering of the *Sce*DIP and *Hsa*HPRD networks by ASModel, joint clustering of the *Sce*DIP and *Sce*BGS networks by AS-Model, as well as results obtained from AignNemo and other single network clustering algorithms. We observe in Fig. 4.11 that joint clustering of the *Sce*DIP and *Sce*BGS networks yields the best F-measure and the largest number of matched reference complexes. However, joint clustering of the *Sce*DIP and *Hsa*HPRD networks achieves the second best F-measure and detects competitive numbers of matched reference complexes as RMCL.

For the *Hsa*HPRD network, we compare the results of ASModel obtained from joint clustering of the *Hsa*HPRD and *Hsa*PIPs networks as well as joint clustering of the *Hsa*HPRD and *Sce*DIP PPI networks, AlignNemo, NCut, MCL, and RMCL. The comparison for the number of matched reference complexes and F-measure is given in Fig. 4.11. From the figure, we find that RMCL gets the best performance in terms of these two metrics. ASModel achieves the competitive performance when joint clustering two human networks as shown before. ASModel for two human networks provides better results than jointly analyzing two networks for yeast and human. From this set of experiments, we find that joint clustering two networks within the same species works better than analyzing networks for different species. We in fact expect this because networks within the same species have more shared information, which can be utilized to supplement each other to improve clustering performance. Otherwise, for two networks for different species, joint clustering may not help as much since they may have different cellular constitution and organization due to evolutionary differences.

GO Enrichment Analysis

We further illustrate the performance comparison for clustering the *Sce*DIP network in Fig. 4.12A. We note that the curve of ASModel for the *Sce*DIP and *Sce*BGS networks is on top of the curve of ASModel for the *Sce*DIP and *Hsa*HPRD PPI networks. Furthermore, both curves from ASModel are on top of all the other algorithms. With respect to the *Hsa*HPRD PPI networks, we have the same observation

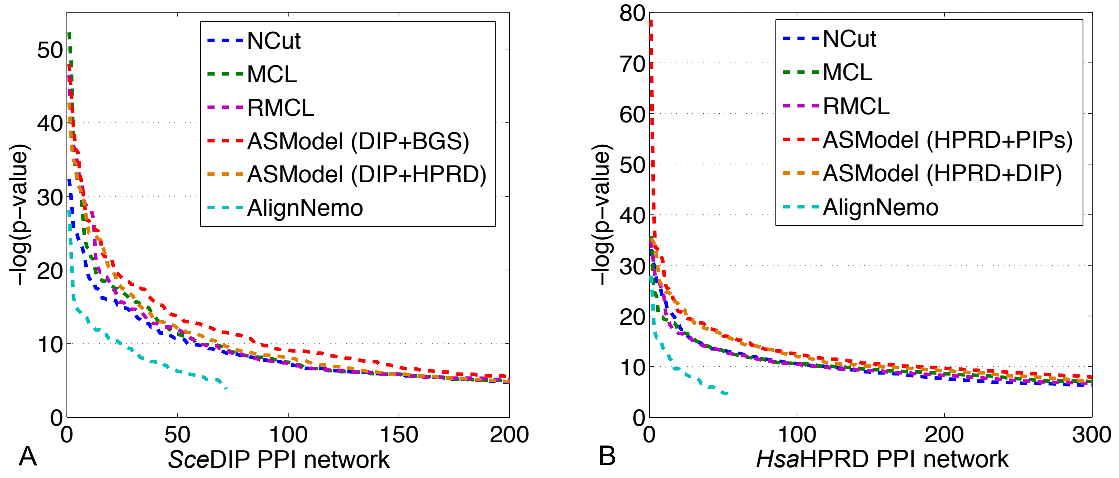


Figure 4.12: Performance comparison of competing algorithms for GO enrichment analysis on the *Sce*DIP and *Hsa*HPRD networks. A. GO enrichment comparison on the *Sce*DIP network. B. GO enrichment comparison on the *Hsa*HPRD network.

that ASModel analyzing PPI networks within the same species is on top of ASModel analyzing networks from different species. Both of them are on top of the others. This further convinces us that joint clustering does improve the clustering performance. In addition, the more information that two PPI networks share, the more enhancement can be achieved by joint clustering. From the comparison of the number of enriched GO terms as shown in Fig. 4.13, we have the same conclusion. ASModel analyzing networks within the same species detects the largest number of enriched GO terms. For analyzing networks from different species, ASModel identifies the second largest number of enriched GO terms among all competing algorithms.

From these experiments, no matter analyzing two PPI networks from the same species or from two different species, our joint clustering algorithm ASModel can achieve better results than analyzing these networks separately using single network clustering algorithms.

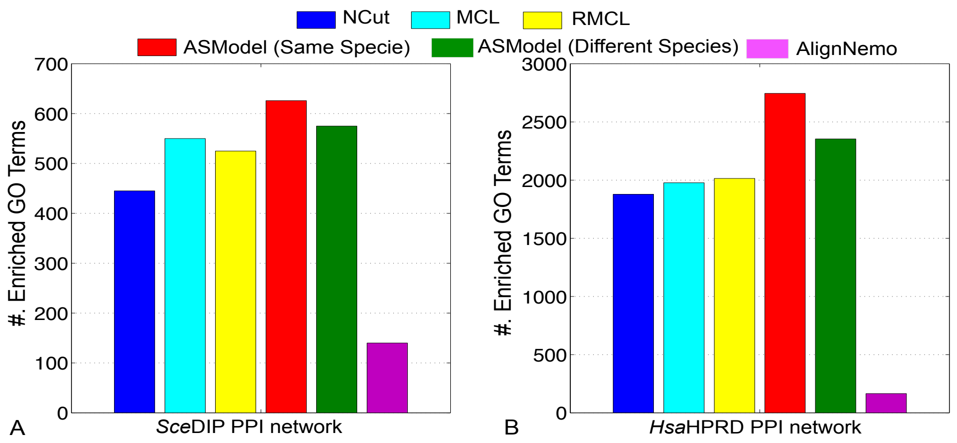


Figure 4.13: Comparison on the number of enriched GO terms for all the competing algorithms in the *Sce* DIP and *Hsa*HPRD networks.

4.2.4 Conclusions

In this section, we have proposed a joint network clustering algorithm ASModel based on a new alternative random walk strategy. The experimental results based on both complex prediction and GO enrichment analysis demonstrate that using ASModel to joint clustering two PPI networks can achieve better clustering results than single network clustering algorithms and AlignNemo. Furthermore, from comparing with the performances of joint clustering PPI networks within the same species and those from different species, we find that more information the PPI networks in the integrated network share, the better the clustering results can be achieved.

5. OVERCOMING THE DEGREE HETEROGENEITY

5.1 Protein complex identification for individual networks

It is well known that protein complexes are densely connected inside and loosely connected outside [69]. Modularity based algorithms have been demonstrated to be effective on identification of protein complexes in yeast and human protein interaction networks [31, 65]. Conductance of a set S of nodes is another widely used definition for protein complex prediction [69], which is defined as the ratio of the number of edges pointing outside the set to the number of edges in the smaller of S and its complementary S^c [6].

However, these methods have their own limitations. Modularity based methods have the resolution problem, which is they can not detect small-size modules [30], therefore protein complexes of small size may be ignored by those methods. Algorithms based on finding low-conductance sets focus on the separability of a subnetwork, which indicates whether the subnetwork is well separated from the rest of the network. However, the internal connections of the subnetwork are not taken into account, which leads to that the densely connected part of the subnetwork can not be distinguished with the presence of the nodes of small degrees.

Degree heterogeneity is a common phenomenon presented in protein interaction networks [4]. Fig. 5.1 shows the degree distribution of the yeast protein interaction network obtained from Database of Interacting Proteins (DIP), which has 5,138 proteins and 22,491 protein-protein interactions. We notice that around half of the proteins in this network have degrees lower than three. Therefore, efforts need to be made to deal with the problem brought by degree heterogeneity of protein interaction networks to discover potential protein complexes.

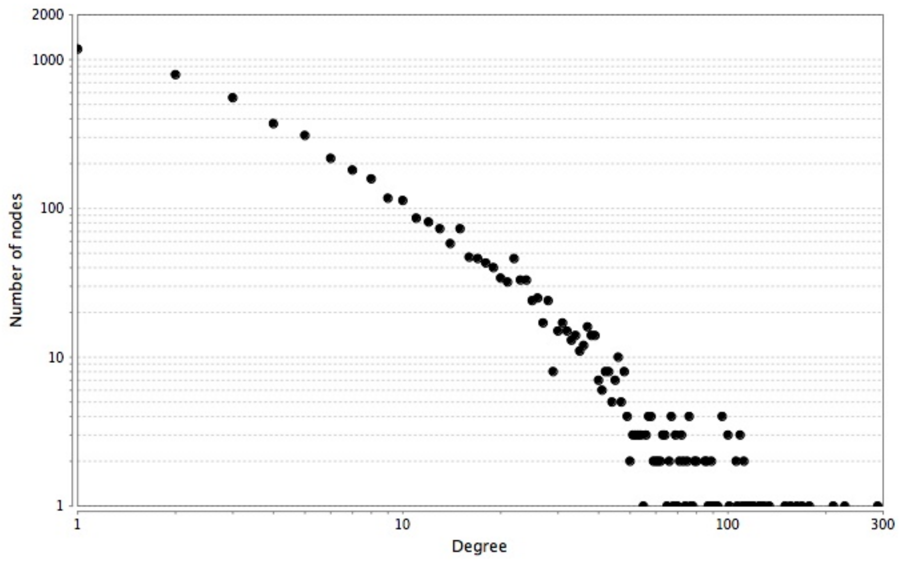


Figure 5.1: Degree distribution of the yeast protein interaction network extracted from DIP database.

In this section, we propose a local protein complex prediction algorithm, which can resolve degree heterogeneity in protein interaction networks. Our FLCD algorithm first identifies a low-conductance set around a node, which is locally well separated from the rest of the network. Then the densely connected part in the set is detected through identifying the densest subgraph within the set, which automatically get rid of the nodes with small degrees. We compare our FLCD with three state-of-the-art overlapping module identification algorithms, which are ClusterONE [69], LinkComm [1] and SR-MCL [98], respectively. Experimental results demonstrate that our FLCD outperforms all competing algorithms on protein complex prediction.

5.1.1 FLCD algorithm

let $G = (V, E)$ represent a protein interaction network, where V denotes the set of nodes and E is the edge set. A is the adjacency matrix of G, of which the element $A_{ij} = 1$ denotes node i interacts with node j and $A_{ij} = 0$ otherwise. The degree matrix D of G is a diagonal matrix with $D_{ii} = d_i$, where d_i is the degree of node i.

For a set S of nodes, the conductance of S is defined as

$$\phi(S) = \frac{|E(S,\bar{S})|}{\min\left\{vol(S), vol(\bar{S})\right\}}, S \cup \bar{S} = V, \tag{5.1}$$

where $E(S,\bar{S})$ denotes the edges between set S and $\bar{S}$ and $vol(T) = \sum_{i \in T} d_i$ is the number of edges of set T. Here we make a mild assumption that $vol(S) << vol(V)$ for large scale protein interaction network G, which means $vol(S) = \min\left\{vol(S), vol(\bar{S})\right\}$. Hence, we have

$$\phi(S) = \frac{|E(S,\bar{S})|}{vol(S)} = \frac{\sum_{ij} D^S_{ij} - A^S_{ij}}{\sum_i D^S_{ii}}, \tag{5.2}$$

where A^S is the adjacency matrix of the induced subnetwork with respect to node set S and D^S is the degree matrix corresponding to A^S, where $D^S_{ii} = d_i, i \in S$. For the same set S, the density of S is defined as

$$\mathcal{D}(S) = \frac{|E(S,S)|}{|S|} = \frac{1}{2}\frac{\sum_{ij} A^S_{ij}}{\sum_{i \in S} 1}. \tag{5.3}$$

5.1.1.1 Searching for a low-conductance set H^*_v near v

Given a starting node v, our goal is to find a node set H^*_v with low conductance including v. We first use the algorithm proposed in [6] to find a set H with potential low conductance, then the exact minimal conductance set H^*_v in H is identified through solving a mixture integer programming problem.

Following [6], a low-conductance set including v can be efficiently obtained via the personalized PageRank vector of v. The personalized PageRank vector $p(\alpha, v)$ of v on G is the stationary distribution of the random walk on G, in which at every step, the random walker has α probability to restart the random walk at v and otherwise performs a lazy random walk. Mathematically, $p(\alpha, v)$ is the unique solution to

$$p(\alpha, v) = \alpha e_v + (1-\alpha) p(\alpha, v) W, \tag{5.4}$$

where $\alpha \in (0, 1]$ is the "teleports" constant, e_v is the indicator vector of v and $W = \frac{1}{2}(I + D^{-1}A)$ is the transition matrix of the lazy random walk. We apply the local algorithm in [6] to efficiently approximate $\hat{p} \approx p(\alpha, v)$. Then we sort the nodes based on $\hat{p}$ and attain a ordering set $\mathcal{H} = \{v_1, v_2, ..., v_n\}$, elements of which satisfy $\hat{p}(v_i) > \hat{p}(v_{i+1})$. We take the top k elements out of $\mathcal{H}$, which are more likely to comprise a low-conductance set with v, and put them in H. The lowest-conductance set H_v^* in H can be computed by solving the following optimization problem based on (5.2).

$$\begin{aligned} \text{min: } & \frac{x^T(D^H - A^H)x}{x^T d^H} \\ s.t. \ & x_i \in \{0, 1\}, \end{aligned} \tag{5.5}$$

where x is a binary vector with $x_i = 1$ indicating that node i is assigned into H_v^* and $x_i = 0$ otherwise and d^H is a vector containing the degrees of every node in H. After some manipulations, (5.5) can be transformed into the following equivalent formulation.

$$\begin{aligned} \text{min: } \ & z \\ s.t. \ \ & z \sum_i x_i d_i^H - \sum_i \sum_j (D_{ij}^H - A_{ij}^H) x_i x_j \geq 0, \\ & x_i \in \{0, 1\}. \end{aligned} \tag{5.6}$$

After using common techniques [29] to linearize zx_i and $x_i x_j$, the optimization problem can be solved by any mixture integer programming solver. Because the size of $|H| = k$ is much smaller than $|V| = n$ and we only focus on identifying one low-conductance set, we can efficiently obtain the lowest-conductance set H_v^* in H based on (5.6).

5.1.1.2 Conservation of the densest subgraph C_v^* in H_v^*

The induced subnetwork G_v with respect to node set H_v^* is well separated from the rest of the network, however, there may exist nodes with low degrees in H_v^*. To remove low-degree nodes as well as reserve densely connected subnetwork, we apply the definition of density (5.3) to find the densest subgraph in H_v^*. Because the problem size is small, we can also make use of the power of mixture integer programming. The node set $C_v^* \in H_v^*$ corresponding to the densest subnetwork can be identified based on (5.3) by

$$\begin{aligned} \text{max:} \quad & \frac{r^T A_{ij}^{H_v^*} r}{r^T \mathbf{1}} \\ s.t. \quad & r_i \in \{0,1\}, \end{aligned} \tag{5.7}$$

where $\mathbf{1}$ is an all one vector and r is the binary matrix indicating the memberships of the nodes in the densest subnetwork. (5.7) can be transformed into

$$\begin{aligned} \text{max:} \quad & w \\ s.t. \quad & w \sum_i r_i - \sum_i \sum_j A_{ij}^{H_v^*} r_i r_j \leq 0, \\ & r_i \in \{0,1\}, \end{aligned} \tag{5.8}$$

which can be casted into mixture integer programming solver after linearization [29].

5.1.1.3 The FLCD algorithm

The step-by-step procedure of FLCD algorithm is shown in Table 5.1. The FLCD algorithm screens every node with degree larger than two. For each selected node, the FLCD algorithm first searches for a low-conductance set around it and then find the densest subnetwork in the low-conductance set, which is considered as a predicted module. After screening every possible node, we remove the duplicated modules and modules with sizes less than three.

Table 5.1: The FLCD algorithm

```
Algorithm: The FLCD Algorithm
Input: $\mathcal{S} = V$ and $k = 30$.
Output: A set of predicted modules $R$.
1  While ($\exists v \in \mathcal{S}$ and $d_v \geq 3$)
2      Estimate $\hat{p} \approx p(\alpha, v)$.
3      Sort nodes in $V$ based on $\hat{p}$ and collect the top $k$ nodes in $H_v$.
4      Finding the lowest-conductance set $H_v^* \in H_v$ based on (5.6).
5      Identifying the node set $C_v^*$ of the densest subnetwork in $H_v^*$ based on (5.8).
6      Considering $C_v^*$ as one predicted module, let $R = \{R, C_v^*\}$ and $\mathcal{S} = \mathcal{S} - v$.
7  EndWhile
8  Remove duplicated modules in $R$.
```

5.1.2 Experimental results

In this section, we compare our FLCD algorithm with other three state-of-the-art overlapping module identification algorithms on protein complex prediction.

5.1.2.1 Data and metric

We use four yeast protein interaction networks SceDIP, SceBG, SceIntAct and SceMINT extracted from DIP [90], BioGrid [101], IntAct [41] and MINT [48], respectively. We remove all genetic interactions for SceBG. We use protein complexes obtained from SGD [37] and MIPS [63] as the golden standard protein complexes. For each protein interaction network, we remove reference protein complexes if their size less than 3 or half of the proteins of them are not in the network. The detailed information of four protein interaction networks and the protein complex golden standards are shown in Table. 5.2.

Table 5.2: The detailed information of four yeast protein interaction networks

Network	#. proteins	#. interactions	SGD	MIPS
SceDIP	5136	22491	224	184
SceBG	6438	80577	234	189
SceIntAct	5453	54134	231	187
SceMINT	5414	27316	230	188

For protein complex prediction, we assess the performance of all competing algorithms by a composite score consisting of three quality scores: F-measure [98]; the geometric accuracy (Acc) score; and the maximum matching ratio (MMR) [69]. For fair comparison, we remove modules of two or less proteins for all competing algorithms.

For a golden standard protein complex set $C = \{c_1, c_2, ..., c_n\}$ and a set of predicted modules $S = \{s_1, s_2, ..., s_m\}$, the F-measure is defined as the harmonic mean of precision and recall.

$$\text{precision} = \frac{|N_{cp}|}{|C|}, \qquad \text{recall} = \frac{|N_{cs}|}{|S|}. \tag{5.9}$$

$N_{cp} = \{c_i \in C | NA(c_i, s_j) \geq 0.25, \exists s_j \in S\}$ is a set of reference protein complexes that are matched by a predicted modules. We consider a reference protein complex c_j is matched by a predicted module s_j if $NA(c_i, s_j) \geq 0.25$ [98], where $NA(c_i, s_j) = \frac{|c_i \cap s_j|^2}{|c_i| \times |s_j|}$ is called neighborhood affinity. $N_{cs} = \{s_i \in S | NA(c_j, s_i) \geq 0.25, \exists c_j \in C\}$ is the set of the modules that match to one or more reference protein complexes.

Finally, the F-measure is

$$\text{F-meature} = 2 \times \frac{\text{precision} * \text{recall}}{\text{precision} \times \text{recall}}. \tag{5.10}$$

The geometric accuracy (Acc) score is the geometric mean of two other measures, which are the cluster-wise sensitivity (Sn) and the cluster-wise positive predictive value (PPV) [69]. Given m predicted and n reference complexes, let t_{ij} denote the number of proteins that exist in both predicted module s_i and reference complex c_j, and w_j represent the number of proteins in reference complex c_j. Then Sn and PPV can be defined as

$$\text{Sn} = \frac{\sum_{j=1}^{n} \underset{i=1,\dots,m}{max}\ t_{ij}}{\sum_{j=1}^{n} w_j}; \qquad \text{PPV} = \frac{\sum_{i=1}^{m} \underset{j=1,\dots,n}{max}\ t_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij}}. \tag{5.11}$$

The Acc score is the balance of Sn and PPV: Acc $= \sqrt{\text{Sn} \times \text{PPV}}$.

The maximum matching ratio [69] is the ratio of the maximum sum of weights of edges in a bipartite graph, where the two sets of nodes are reference complexes C and predicted complexes S, to the number of reference protein complexes $|C|$. The bipartite graph is represented by a weighted matrix $B_{n\times m}$, where the edge weight B_{ij} between nodes c_i and s_j is the neighborhood affinity score $NA(c_i, s_j)$. The MMR is the solution of the following maximal matching problem.

$$\begin{aligned} \text{max:} \quad & \frac{1}{|C|} \sum_{i=1}^{n} \sum_{j=1}^{m} B_{ij} \sigma_{c_i,s_j} \\ s.t. \quad & \sum_{j=1}^{m} \sigma_{c_i,s_j} \leq 1 \\ & \sum_{i=1}^{n} \sigma_{c_i,s_j} \leq 1, \end{aligned} \tag{5.12}$$

where σ is an indicator function. $\sigma_{c_i,s_j} = 1$ when the edge between nodes c_i and s_j is selected and $\sigma_{c_i,s_j} = 0$ otherwise.

5.1.2.2 Protein complex prediction

We compare all competing algorithms in terms of the composite score, consisting of F-measure, Acc score and MMR. For SGD protein complex dataset, the detailed comparison results are shown in Table. 5.3. As shown in the table, our FLCD consistently achieve the best F-measure and MMR score and the Acc score is competitive to LinkCommunity, which achieves the best Acc score for all four yeast protein interaction networks. But for the composite score, as shown in Fig. 5.2 our FLCD outperforms all other state-of-the-art algorithms, which indicates that FLCD has the best performance for predicting protein complexes in SGD dataset.

Additionally, we make comparison on prediction for MIPS protein complex golden standard set. Table. 5.4 displays the detailed scores. We find the same trend for the F-measure and MMR scores, which is our FLCD attains the best F-measure and MMR scores for all four yeast protein interaction networks. For Acc score, FLCD has the best Acc scores for SceBG and SceIntAct, but LinkComm obtain the best Acc scores for SceDIP and SceMINT. However, the overall performance, which is represented by the composite score, of FLCD is superior to other competing algorithms as shown in Fig. 5.3.

5.2 Discover conserved protein complexes in multiple networks

In this section, we extend the idea of FLCD to multiple networks, which we called ClusterM.

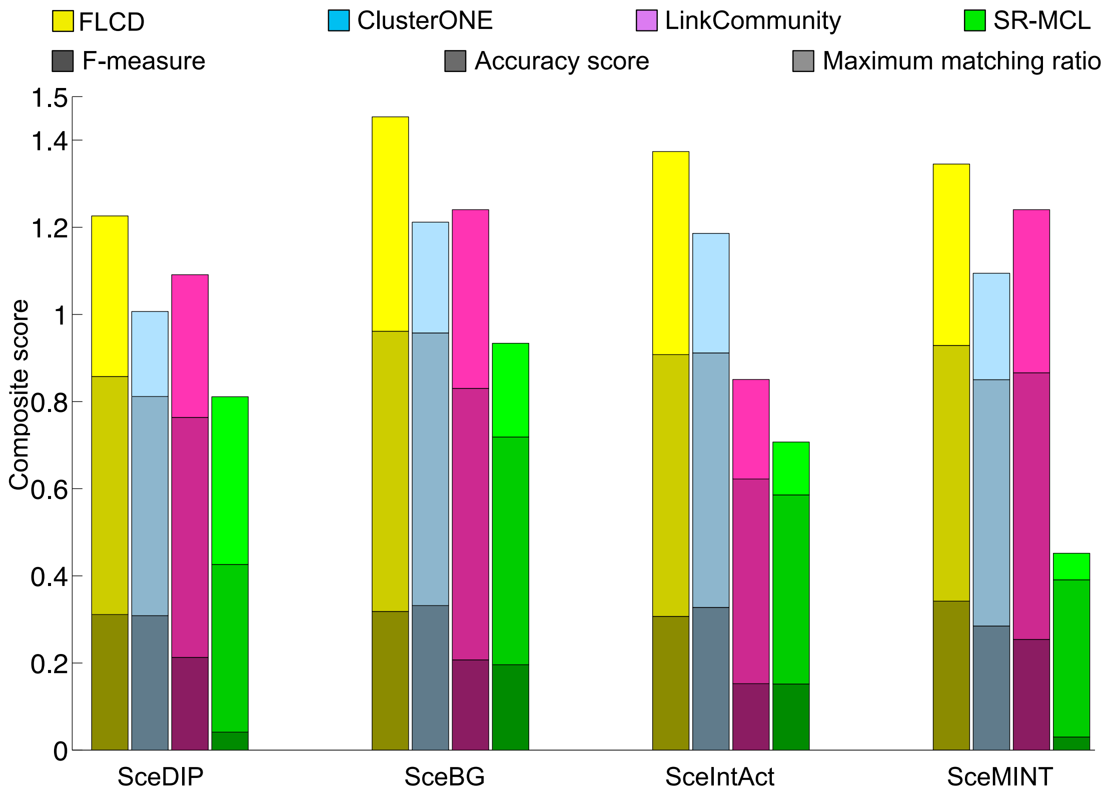


Figure 5.2: Comparison among all competing algorithms on SGD dataset in terms of the composite scores. CONE and LinkC are short for ClusterONE and LinkComm

Table 5.3: Comparison of protein complex prediction on SGD dataset.

Network	method	# modules	#. matched	F-measure	Sn	PPV	Acc	MMR
SceDIP	FLCD	2134	**152**	**0.3113**	0.5964	0.5003	0.5462	**0.3685**
	CONE	380	86	0.3085	0.4082	**0.6203**	0.5032	0.1950
	LinkC	1839	137	0.2130	**0.6290**	0.4820	**0.5506**	0.3276
	SR-MCL	2851	44	0.0412	0.5120	0.2893	0.3489	0.0708
SceBG	FLCD	4027	**183**	**0.3181**	0.7363	0.5621	**0.6433**	**0.4920**
	CONE	522	122	0.3318	0.6488	**0.6035**	0.6257	0.2542
	LinkC	5382	164	0.2072	0.8880	0.4373	0.6231	0.4100
	SR-MCL	1862	108	0.1961	**0.8999**	0.3034	0.5225	0.2151
SceIntAct	FLCD	3394	**172**	**0.3069**	0.6699	0.5391	**0.6009**	**0.4661**
	CONE	496	117	0.3275	0.5742	**0.5944**	0.5842	0.2742
	LinkC	1297	93	0.1525	**0.9223**	0.2393	0.4698	0.2285
	SR-MCL	1079	68	0.1517	0.7784	0.2402	0.4341	0.1213
SceMINT	FLCD	2483	**157**	**0.3418**	0.6524	0.5284	0.5871	**0.4163**
	CONE	513	110	0.2848	0.5370	**0.5954**	0.5654	0.2442
	LinkC	3698	144	0.2542	**0.6757**	0.5540	**0.6119**	0.3743
	SR-MCL	2201	33	0.0302	0.5013	0.2597	0.3608	0.0609

CONE and LinkC are short for ClusterONE and LinkComm.

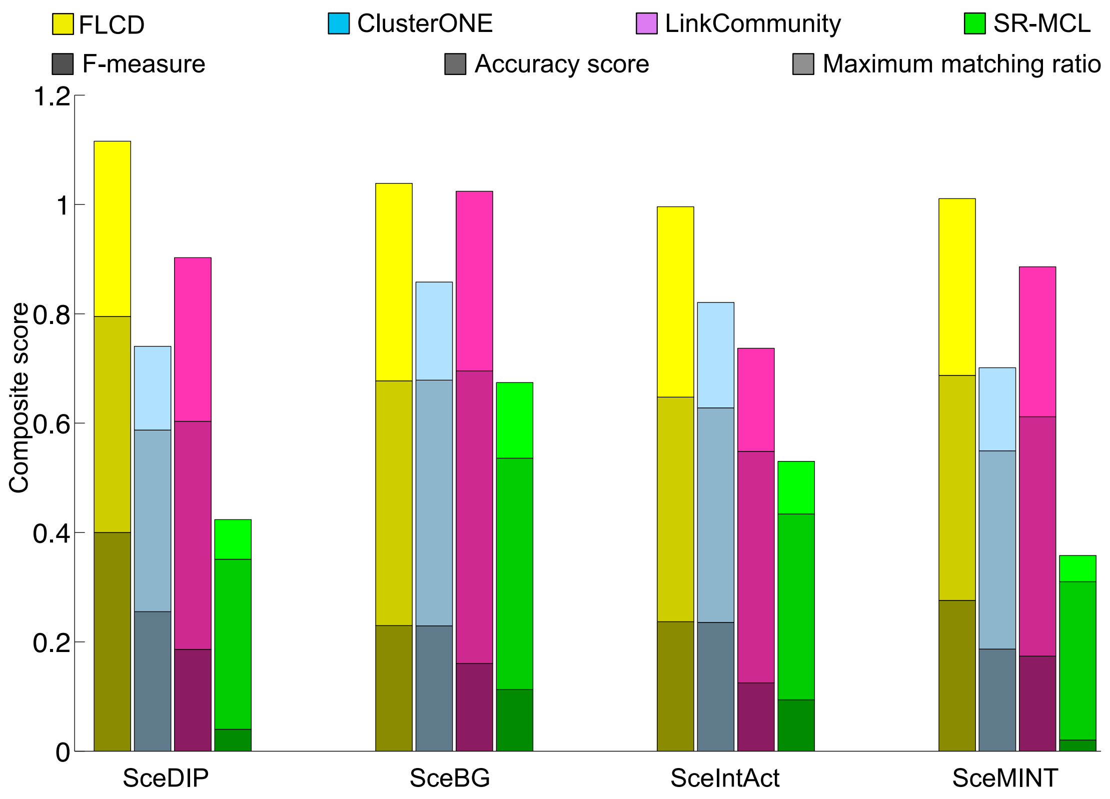


Figure 5.3: Comparison among all competing algorithms on MIPS dataset in terms of the composite scores. CONE and LinkC are short for ClusterONE and LinkComm.

Table 5.4: Comparison of protein complex prediction on MIPS dataset.

Network	method	# modules	#. matched	F-measure	Sn	PPV	Acc	MMR
SceDIP	FLCD	2134	**120**	**0.2573**	0.4001	0.3901	0.3951	**0.3206**
	CONE	380	74	0.2551	0.2749	**0.4015**	0.3322	0.1533
	LinkC	1839	109	0.1862	**0.4775**	0.3646	**0.4173**	0.2993
	SR-MCL	2851	41	0.0402	0.4592	0.2104	0.3108	0.0726
SceBG	FLCD	4027	**124**	**0.2298**	0.4643	0.4315	0.4476	**0.3611**
	CONE	522	86	0.2293	0.4537	**0.4452**	0.4494	0.1795
	LinkC	5382	109	0.1604	**0.8179**	0.3504	**0.5354**	0.3285
	SR-MCL	1862	65	0.1126	0.7360	0.2436	0.4234	0.1384
SceIntAct	FLCD	3394	**120**	**0.2368**	0.4183	0.4034	0.4108	**0.3482**
	CONE	496	79	0.2356	0.3587	**0.4296**	0.3925	0.1927
	LinkC	1297	80	0.1251	**0.9028**	0.1986	**0.4234**	0.1886
	SR-MCL	1079	45	0.0941	0.6246	0.1850	0.3399	0.0960
SceMINT	FLCD	2483	**111**	**0.2759**	0.4147	**0.4086**	0.4116	**0.3231**
	CONE	513	67	0.1869	0.3274	0.4017	0.3626	0.1519
	LinkC	3698	100	0.1740	**0.4744**	0.4038	**0.4377**	0.2744
	SR-MCL	2201	24	0.0205	0.4192	0.1999	0.2894	0.0481

CONE and LinkC are short for ClusterONE and LinkComm.

5.2.1 ClusterM

Our ClusterM algorithm builds on the intuition that functional conserved modules should possess the following topological and homologicial properties simultaneously. Topologically, the conserved module in each protein interaction network is well separated from the rest of the network in order to give rise to a unique and specific biological form and function. Additionally, the network structure of the conserved module is dense clusters of interactions, which models protein complexes. Homologicially, there should exist a many-to-many correspondence between the proteins of the modules conserved in different protein interaction networks, so the number of the sequence-similar protein pairs between modules across networks has to be as many as possible.

To identify conserved modules with the above properties, briefly, our ClusterM algorithm has three steps. In the first step, potential orthologous proteins are identified by a network alignment method, which takes into account both protein interaction and sequence information. A group of orthologous proteins, which consists exact one protein from one network, is defined as a *spine*.

Given k protein interaction networks $G = \{G_1, G_2, ...G_k\}$, where $G_i(V_i, E_i)$ is the ith network with V_i and E_i representing the corresponding nodes and edges respectively, the orthologous protein *spines* can be identified by any multiple network alignment methods, which extract proteins from different networks based on their sequences and neighborhood topologies. Here we use SMETANA [89] to attain h potential protein *spines* $\mathcal{U} = \{u^1, u^2, ..., u^h\}$, where $u^i = \{(v^i_1, v^i_2, ..., v^i_k) | v^i_1 \in V_1, v^i_2 \in V_2, ..., v^i_k \in V_k\}$ is the ith protein *spine*.

In the second step, the local network structure of each protein in every *spine* is investigated to obtain a subnetwork, which is well separated from the rest of the network. The separability of an induced subnetwork G_b in G is characterized by conductance, the definition of which is the ratio of the number of edges connecting from G_b to the nodes in $\bar{G}_b$, which is the remaining part of the network after removing G_b, to the total number of edges in the smaller of these G_b and $\bar{G}_b$ subnetworks.

For protein v_i^j in protein *spine* u^j, a local search starting from v_i^j is performed in G_i to identify an induced subnetwork including v_i^j with low conductance. The initial search is guided by a reliable estimate of the personalized PageRank vector of v_i^j [6]. We then take out m proteins with the top-ranking scores in the personalized PageRank vector and further refine the result by solving a mixture integer programming problem [29], which yields a subnetwork $G_i^j(V_i^j, E_i^j)$ with the lowest conductance. The same procedure is repeated until we obtain the set of subnetworks $G^j = \{G_1^j, G_2^j, ..., G_k^j\}$ around every protein in the protein *spine* u^j.

In the final step, we screen every protein *spines* to find functional conserved modules. For protein *spine* u^j, we integrate all the subnetworks G^j corresponding to it and the sequence information of the proteins within those subnetworks to identify the conserved modules with respect to u^j. In order to detect functional conserved modules with sufficient topological connections as well as strong homological correspondence, we define the following cost function for the protein *spine* u^j

$$\mathcal{F}^j = \sum_{i=1}^{k} \frac{\sum_{s,t \in V_i^j} A_i^j(s,t)\sigma_s\sigma_t}{\sum_{s \in V_i^j} \sigma_s} + \lambda \frac{\sum_{k,l} \sum_{(p,q) \in O(V_k^j, V_l^j)} s(p,q)\sigma_p\sigma_q}{\sum_{i=1}^{k} \sum_{s \in V_i^j} \sigma_s}. \tag{5.13}$$

Here, $\{V_1^j, V_2^j, ..., V_k^j\}$ are the set of proteins for $G^j = \{G_1^j, G_2^j, ..., G_k^j\}$ corresponding to *spine* u^j, respectively. A_i^j is the unweighted adjacency matrix of subnetwork G_i^j with $A_i^j(s,t) = 1$ denoting the interaction between proteins s and t in G_i^j and $A_i^j(s,t) = 0$, otherwise. The value of σ_s equals one if protein s is identified to be in the conserved modules and zero otherwise. The first term of the cost function $\mathcal{F}^j$ is essentially the summation of the density functions of k individual subnetworks. The density function can not only help us to remove proteins with few interactions in the subnetwork, but also enable us to preserve the functional modules after evolution (examples are in section 5.2.1.2). The last term computes the density of similarity, which is the ratio of the summation of sequence similarities between proteins across the subnetworks to the total number of proteins selected in the final conserved modules over all subnetworks. $O(V_k^j, V_l^j)$ denotes all the sequence-similar protein pairs between V_k^j and V_l^j, which is weighted by the normalized bit score

$$s(p,q) = \frac{blast(p,q)}{\sqrt{blast(p,p) \times blast(q,q)}}, \tag{5.14}$$

where $blast(p,q)$ is the bit score of the sequence similarity between proteins $p \in V_k^j$ and $q \in V_l^j$ calculated by the local sequence alignment tool BLAST [121]. The relative contributions between the local topological structures and the homological correspondences are controlled by a coupling constant λ.

We optimize the cost function (5.13) in a greedy manner. Initially, we recruit all the proteins $V^j = \bigcup_i V_i^j$ in subnetwork set G^j, and remove a protein with the most impairment to the cost function at each iteration until further deletion would not benefit the cost function.

5.2.1.1 A local algorithm for searching a cohesive subnetwork

We propose a local algorithm for searching a cohesive subnetwork staring from a specific node v in $G(V,E)$, which is based on finding a low-conductance set containing v. The conductance $\Phi(S)$ of a set S in G is defined as the ratio of the number of edges between S and its complementary set $\bar{S}$ ($S \cup \bar{S} = V$) to the number of edges in the larger of those two sets. Obviously, based on the definition, the nodes in the lowest-conductance set highly interact with the nodes inside rather than the nodes outside, which provides us a natural measure of separability of a subset of the nodes in G. Formally,

$$\Phi(S, \bar{S}) = \frac{|\partial S|}{\min\big(d(S), d(\bar{S})\big)}, \tag{5.15}$$

where $\partial S = \{(v_i, v_j)|v_i \in S, v_j \in \bar{S}\}$ and $d(S)$ is the sum of the degrees of nodes in S.

There are two sub-tasks of our local algorithm: (i) estimating a low-conductance set S_v containing v of size $|S_v| = m$ (ii) searching the lowest conductance set $S_v^* \subseteq S_v$ by the mixture integer programming.

Estimation of a low-conductance set S_v

[6] shows that a low-conductance set containing v can be efficiently estimated via the personalized PageRank vector of v. The personalized PageRank vector of v is an invariant distribution of a lazy random walk on G with restart. The transition probability matrix of the lazy random walk is $W = \frac{1}{2}(I + D^{-1}A)$, where D is a diagonal matrix with the degrees of nodes on its diagonal. At each step of the random walk, the random walker has α probability to restart to walk on node v and $1 - \alpha$ probability to do the lazy random walk. Thus, the personalized PageRank vector of v is the unique solution of

$$p(v) = \alpha e_i + (1 - \alpha)p(v)W, \tag{5.16}$$

where $\alpha \in (0, 1]$ is the teleportation constant and e_i is an all zeros vector except the ith entry is 1. We use an efficient algorithm proposed in [6] to approximate $\hat{p} :\approx p(v)$. Given $\hat{p}$, the estimation of $p(v)$, let $v_1, v_2, ..., v_{|V|}$ be an ordering of nodes such that $\hat{p}(v_i) \leq \hat{p}(v_{i+1})$. We take the first m nodes as a low-conductance set

$S_v = \{v_1, v_2, ..., v_m\}$. From [6], we know the starting node $v \in S_v$ and the lowest conductance in S_v has a upper bound guarantee.

Searching the lowest-conductance set in S_v

We use mixture integer programming to assist us to find the minimal conductance set $S_v^* \subseteq S_v$. Biologically, the size of a functional module is much smaller than the size of a protein interaction network, therefore, we make a reasonable assumption that is $d(S_v) << d(\bar{S}_v)$. The definition of conductance becomes

$$\Phi(S, \bar{S}) = \frac{|\partial S|}{d(S)}. \tag{5.17}$$

We know that $\Phi(S, \bar{S}) + \Phi(S, S) = 1$ from the lemma in the supplementary materials of [110]. Thus, finding the lowest-conductance set in S_v is equivalent to

$$S_v^* := arg \min_{v \in S_v^t \subseteq S_v}: \Phi(S_v^t, \bar{S_v^t}) \Leftrightarrow S_v^* := arg \max_{v \in S_v^t \subseteq S_v}: \Phi(S_v^t, S_v^t). \tag{5.18}$$

Furthermore, the above problem can be written in a closed form.

$$\begin{aligned} \max_x: \quad & \frac{x^T A^{S_v} x}{x^T d^{S_v}} \\ s.t. \quad & x(v) = 1, \\ & x(v_i) \in \{0, 1\}, \forall v_i \neq v \end{aligned} \tag{5.19}$$

where A^{S_v} is the adjacency matrix of the induce subnetwork of S_v and d^{S_v} is the vector of the degrees of nodes S_v in the G. x is a binary vector, the ith entry of which indicates node v_i is in the lowest-conductance set if $x(v_i) = 1$. The constraint $x(v) = 1$ ensures that node v is included in the lowest-conductance set. (5.19) can be transformed in to a mixture integer programming problem [29] as following.

$$\begin{aligned} \text{max:}\ \ & z \\ s.t.\ \ & z\left(\sum_{i\in S_v} x(i)d_i^{S_v}\right) - \sum_{i\in S_v}\sum_{j\in S_v} A_{ij}^{S_v} x(i)x(j) \le 0, \\ & x(v) = 1, \\ & x(i) \in \{0,1\},\ \forall i \neq v, \end{aligned} \tag{5.20}$$

where we assume $z = \dfrac{x^T A^{S_v} x}{x^T d^{S_v}}$ and use the fact that $x^T A^{S_v} x = \sum_{i\in S_v}\sum_{j\in S_v} A_{ij}^{S_v} x(i)x(j)$ and $x^T d^{S_v} = \sum_{i\in S_v} x(i)d_i^{S_v}$. The product of $zx(i)$ and $x(i)x(j)$ can be linearized [29]. Therefore, (5.20) is transformed into a mixture integer programming problem, which is solved by Gurobi [34].

5.2.1.2 The density function

The conductance only measures the relationship between external connections and internal connections, therefore the connectivity inside the low-conductance set may not be dense. For example, as shown in Fig. 5.4, the conductance of the subnetwork G is $\dfrac{2}{11}$, which is the lowest, however, obviously the nodes inside G are not well connected. Therefore, we need another criterion to determine the internal structure of the subnetwork.

Here we use the density function. Given an undirected network $G(V, E)$, which is carved out from the original network, then the density function of a subnetwork $G_b(V_b, E_b)$ in G is defined as

$$D_s(G_b) = \frac{|E_s|}{|V_s|} \tag{5.21}$$

If we use the adjacency matrix A to represent the network G and an indicator function

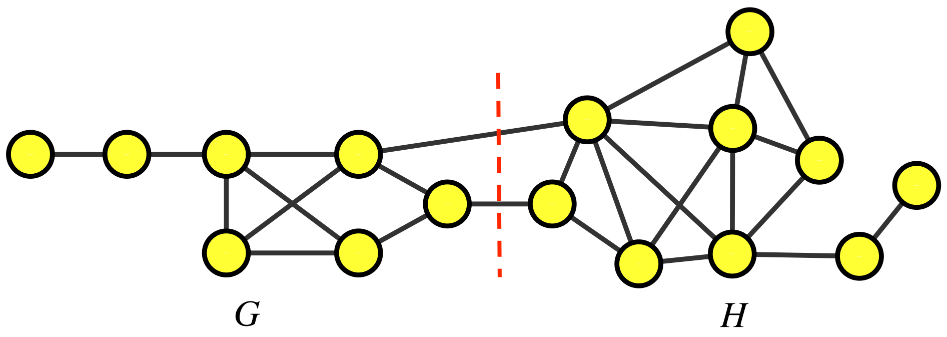


Figure 5.4: An example of a subnetwork with low conductance. The red dash line indicates the network is separated into G and H two parts.

σ_i^b

$$\sigma_i^b = \begin{cases} 1 & i \in V_b \\ 0 & o.w. \end{cases} \tag{5.22}$$

to suggest whether the node i is in V_b, then the density function of G_b can be written as

$$D_s(G_b) = \frac{\sum_{i,j\in V} A_{ij}\sigma_i^b\sigma_j^b}{\sum_{i\in V}\sigma_i^b}. \tag{5.23}$$

Furthermore, we use $D_s^*(G)$ to present the highest density function value of network G. Hence, $D_s^*(G) = D_s(G_b)$ means that G_b is the densest subnetwork in G. In the following part of this section, we assume all the networks in the examples have already been extracted from the original network based on conductance.

First of all, the density function can enable us to remove proteins with less incident edges. As illustrated in Fig. 5.5, the densest subnetwork of G is the subnetwork G' in the shade with $D_s^*(G) = D_s(G') = \frac{7}{5}$, which demonstrates that using density function we can get rid of nodes with small degrees.

Furthermore, the network structures, such as linear paths and densely connected networks, can be characterized by the density function. Fig. 5.6 illustrates two examples. Fig. 5.6(a) is a linear path G_l, the densest subgraph of which is itself with

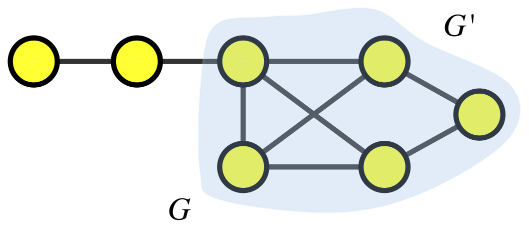


Figure 5.5: A induced subnetwork G. The shade part of G is the subnetwork G'.

$D_s^*(G_l) = D_s(G_l) = \frac{3}{4}$. Similarly, the densest subgraph of G_s shown in Fig. 5.6(b) is also itself with $D_s^*(G_s) = D_s(G_s) = \frac{7}{5}$. The linear path and dense graph can be used to model signal transaction pathways and protein complexes, therefore, it is obvious that the density function (5.23) provides us a powerful tool to distinguish those two kinds of structures, which are of biological significance.

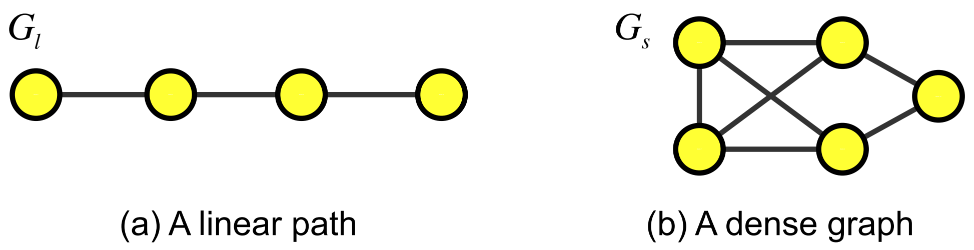


Figure 5.6: Densest subgraph examples. (a) a linear path G_l; (b) a dense graph G_s.

Last but not least, the density function can help us to identify functional modules even after evolution. For example, in Fig. 5.7, we notice that the protein complex G_1 with three proteins can evolve to G_1' and G_1'' based on the duplication/divergence model [46]. Obviously, G_1 itself is a densest network based on (5.23). Also, G_1' and G_1'' can be detected by using the density function (5.23), because the densest subnetwork are themselves ($D_s^*(G_1') = D_s(G_1') = \frac{5}{4}$ and $D_s^*(G_1'') = D_s(G_1'') = \frac{5}{4}$).

Therefore, we find that the protein complexes before and after evolution can all be identified by searching the densest the subnetwork.

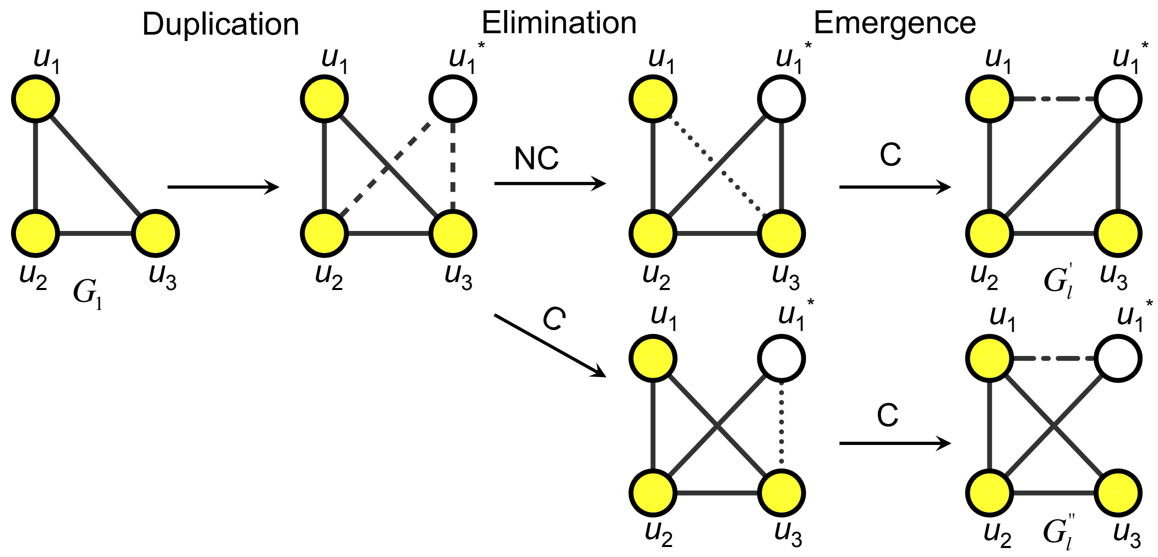


Figure 5.7: Duplication/divergence model for evolution of protein interaction networks. Starting from a protein complex G_1 with three proteins, after the duplication, elimination and emergence process, G_1 evolves to G_1' and G_1''. "NC" denotes the elimination or emergence of edges is uncorrelated to the duplicated node u_1^*. "C" denotes the elimination or emergence of edges is correlated to the duplicated node u_1^*. The dash lines represent the duplicated edges. The dot lines represent the eliminated edges. And the dot dash lines represent the emerged edges.

5.2.2 Experimental results

5.2.2.1 Test data

We apply our algorithm ClusterM to identify conserved modules to two sets of protein interaction networks. In the first set of protein interaction networks, there are four yeast protein interaction networks collected from four public database, which are Database of Interaction (downloaded on January 2015) [90], Biological General Repository for Interaction Datasets (version 3.2.120 downloaded on December 2014) [17], IntAct Molecular Interaction Database (downloaded on January 2015) [41] and Molecular INTeraction database (extracted from IntAct) [48], respectively. We use SceDIP, SceBG, SceIntAct and SceMINT to present the yeast protein interaction networks extracted from Database of Interaction (DIP), Biological General Repository for Interaction Datasets (BioGrid), IntAct Molecular Interaction Database (IntAct) and Molecular INTeraction database (MINT), respectively. The detailed information about this set of protein interaction networks are display in Table. 5.5. The second set of protein interact networks are yeast (*Saccharonyces cerevisiae*), human (*Homo sapiens*), fly(*Drosophila melanogaster*) and worm (*Caenorhabditis elegans*) protein interaction networks obtained from DIP [90] (downloaded on January 2015). The information of the second set of protein interaction networks is shown in Table. 5.6. The protein sequence similarities among proteins are computed by BLAST [121].

Table 5.5: The detailed information for four yeast protein interaction networks.

Network	#. proteins	#. interactions
SceDIP	5136	22491
SceBG	6438	80577
SceIntAct	5453	54134
SceMINT	5414	27316

Table 5.6: The detailed information for four protein interaction networks in DIP database.

Network	#. proteins	#. interactions
SceDIP	5136	22491
HsaDIP	4278	6446
DmeDIP	7679	23182
CelDIP	2712	4117

To validate the capacity of protein complex prediction of our algorithm ClusterM and other state-of-the-art algorithms, we compare the modules yielded by all competing algorithms with the protein complex golden standards. For yeast protein interaction networks, we use MIPS [63] and SGD [37] golden standards. For human protein interaction network, we use CORUM (Comprehensive Resource of Mammalian protein complexes) [87] golden standard. We use all golden standard protein complexes with two or more proteins in all our experiments.

We evaluate whether the proteins in the conserved modules, which span every involved protein interaction network, share similar functions based on Gene Ontology (GO) terms [7] in all three domains (molecular function (F), biological process (P) and cellular component (C)). We only consider the high-level GO terms, which suggest more specific biological meanings than, for example, root GO terms, with information content larger than two. The definition of the information content of a GO term g is $IC = -\log(|g|/|root|)$ [98], where "root" is the corresponding root GO term (either F, P or C) of g and the operation $||$ counts the number of proteins annotated to a specified GO terms.

5.2.2.2 Competing algorithms

We implement ClusterM in Matlab, which supports the multicore parallelism. ClusterM only has one tuning parameter λ that controls the contributions between network structures and sequence similarities in the cost function that serves as the guide for the many-to-many alignments. We compared ClusterM with NetworkBlast(NB) [94], MaWISh [46], NetworkBlast(NB-M)-M [39] and OrthoClust(OC) [116] on every pairwise alignment of the two sets of multiple protein interaction networks and compared with NetworkBlast-M [39] and OrthoClust [116] on alignments with more than two networks. We set the parameters of NetworkBlast [94], MaWISH [46], NetworkBlast-M [39] to their default values. For the parameter κ of OrthoClust, which is also a tradeoff between topology and homology, we applied grid search from $\kappa = 0$ to 50 with interval of 5 and use the results with the maximal objective function value.

We also compare the performance of ClusterM with overlapping module identification algorithms for single networks. ClusterONE [69] and LinkComm [1] were implemented on both yeast and human protein interaction networks. The parameter of LinkComm, which is the threshold for cutting the dendrogram of the hierarchical clustering, was set to 0.2 based on our empirical experiences.

5.2.2.3 Evaluation metrics

We compare ClusterM with all competing algorithms on three aspects. For conserved modules in individual networks, we examine whether the identified modules match to known protein complexes. For aligned conserved modules, proteins of which span every involved protein interaction network, we investigate whether the proteins in the aligned conserved modules perform similar functions. Additionally, the ability of predicting functions of unknown proteins of each algorithm is studied.

Metrics for protein complex prediction

For protein complex prediction, we assess the performance of all competing algorithms by a composite score consisting of three quality scores: F-measure [98]; the geometric accuracy (Acc) score; and the maximum matching ratio (MMR) [69]. For fair comparison, we remove modules of two or less proteins for all competing algorithms.

For a golden standard protein complex set $C = \{c_1, c_2, ..., c_n\}$ and a set of predicted modules $S = \{s_1, s_2, ..., s_m\}$, the F-measure is defined as the harmonic mean of precision and recall.

$$\text{precision} = \frac{|N_{cp}|}{|C|}, \qquad \text{recall} = \frac{|N_{cs}|}{|S|}. \tag{5.24}$$

$N_{cp} = \{c_i \in C | NA(c_i, s_j) \geq 0.25, \exists s_j \in S\}$ is a set of reference protein complexes that are matched by a predicted modules. We consider that a reference protein complex c_j is matched by a predicted modules s_j if $NA(c_i, s_j) \geq 0.25$ [69, 98], where $NA(c_i, s_j) = \dfrac{|c_i \cap s_j|^2}{|c_i| \times |s_j|}$ is called neighborhood affinity. $N_{cs} = \{s_i \in S | NA(c_j, s_i) \geq 0.25, \exists c_j \in C\}$ is the set of the modules that match to one or more reference protein complexes. Finally, the F-measure is

$$\text{F-meature} = 2 \times \frac{\text{precision} * \text{recall}}{\text{precision} \times \text{recall}}. \tag{5.25}$$

The geometric accuracy (Acc) score is the geometric mean of two other measures, which are the cluster-wise sensitivity (Sn) and the cluster-wise positive predictive value (PPV) [69]. Given m predicted and n reference complexes, let t_{ij} denote the number of proteins that exist in both predicted module i and reference complex j, and w_j represent the number of proteins in reference complex j. Then Sn and PPV can be defined as

$$\text{Sn} = \frac{\sum_{j=1}^{n} \underset{i=1,\ldots,m}{max}\ t_{ij}}{\sum_{j=1}^{n} w_j}; \qquad \text{PPV} = \frac{\sum_{i=1}^{m} \underset{j=1,\ldots,n}{max}\ t_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij}}. \tag{5.26}$$

The Acc score is the balance of Sn and PPV: $\text{Acc} = \sqrt{\text{Sn} \times \text{PPV}}$.

The maximum matching ratio [69] is the maximum sum of weights of edges in a bipartite graph, where the two set of nodes are reference complexes C and predicted complexes S. The bipartite graph is represented by a weighted matrix $B_{n\times m}$, where the edge weight B_{ij} between nodes c_i and s_j is the neighborhood affinity score

$NA(c_i, s_j)$. The MMR is the solution of the following maximal matching problem.

$$\begin{aligned} \text{max:} \quad & \frac{1}{|C|}\sum_{i=1}^{n}\sum_{j=1}^{m} B_{ij}\sigma_{c_i,s_j} \\ s.t. \quad & \sum_{j=1}^{m}\sigma_{c_i,s_j} \leq 1 \\ & \sum_{i=1}^{n}\sigma_{c_i,s_j} \leq 1, \end{aligned} \tag{5.27}$$

where σ is an indicator function. $\sigma_{c_i,s_j} = 1$ when the edge between nodes c_i and s_j is selected and $\sigma_{c_i,s_j} = 0$ otherwise.

Metrics for consistency and coverage

We measure the functional consistency for the proteins in the aligned conserved modules by computing the normalized mean entropy (MNE) [54, 99]. We use the GO terms set F to annotate each protein in an aligned conserved module R_i. The union of GO terms used for R_i is $F_i = \{f_1, f_2, ..., f_d\}$. The normalized entropy (NE) of R_i is computed as

$$\mathrm{NE}(R_i) = \mathrm{NE}(p_1, p_2, ..., p_d) = -\frac{1}{\log d}\sum_j p_j \log p_j, \tag{5.28}$$

where p_i is the fraction of R_i with respect to GO term f_i. The MNE is the mean over all $\mathrm{NE}(R_i)$. For the coverage, we simply count the number of proteins in all aligned conserved modules, each of which consists of at least one protein from each network.

Metrics for protein function prediction

Furthermore, we investigate the performance of protein function prediction based on the aligned conversed modules. For protein r_i in the aligned conserved module $R = \{r_1, r_2, ..., r_l\}$, assuming we do not know its functions, if half of the remained proteins in R except r_i share the same functions, we predict that r_i carries out the same functions. Because the aligned conserved modules may overlap and one aligned conserved module may perform multiple functions, r_i may be predicted to have a few functions annotated by a set of high-level GO terms F_{r_i}. We compare F_{r_i} with its true high-level GO term annotations $F^*_{r_i}$. If the intersection of F_{r_i} and $F^*_{r_i}$ is not empty, we consider the function of protein r_i is correctly predicted. The prediction accuracy is the ratio of the number of proteins that are correctly predicted to the total number of proteins, whose functions are predicted.

5.2.2.4 Protein complex prediction

We assess the quality of the conserved modules identified in individual networks through comparisons with the golden standard protein complexes. We collect two sets of referenced protein complexes of yeast from MIPS [63] and SGD [37] and one set of protein-complex golden standard of human from CORUM [87].

Conserved module identification of four yeast protein interaction networks

We applied ClusterM, OrthoClust and NetworkBlast-M to identify conserved modules of protein interaction networks of SceDIP (yeast PIN obtained from DIP database), SceBioGrid (yeast PIN obtained from BioGrid database), SceIntAct (yeast PIN obtained from IntAct database) and SceMINT (yeast PIN obtained from MINT database). Because all these four networks are yeast protein interaction networks and we have the MIPS and SGD protein complex golden standard, the performance of each algorithm on protein complex prediction in each network can be easily verified. Furthermore, we add two state-of-the-art algorithms ClusterONE [69] and LinkComm [1], which can detect overlapping functional modules for individual networks, into the comparison.

Fig. 5.8 shows the comparison of all competing algorithms on SGD golden standard in terms of the composite score. We observe that our algorithm ClusterM with three different selections of λ outperforms all the competing algorithms including two state-out-the-art single network module identification algorithms, which demonstrates that our ClusterM is superior to other algorithms in recovering protein complexes. Furthermore, we find that with the increasing λ, the performance of ClusterM get improved, which makes sense because sequence information between yeast protein interaction networks allows us to find the true correspondences across networks. Therefore, larger λ, indicating to rely more on sequence information than network structures, leads us to achieve better results.

Table 5.7 illustrates the detailed results of various algorithms corresponding to SGD data set for all protein interaction networks. ClusterM consistently achieves the best accuracy score (Acc score) and the maximal matching ratio (MMR), which demonstrates that the predicted modules identified by ClusterM are more similar to the SGD reference protein complexes than modules detected by other methods. In comparison with different selections of λ, we find that ClusterM with larger λ tends to attain higher F-measure score. The reason is that large λ forces ClusterM to neglect sequence-similar proteins across networks if they are not locally densely connected within their networks and, in contrast, only proteins of sequence similarity as well as topological cohesiveness are conserved. λ only has slight influence on the Acc score and MMR.

Fig. 5.9 displays the comparison results between all competing algorithms on MIPS protein complex set. We notice that, similar to the results on SGD date set, three implementations of ClusterM with different λ attain better results than other algorithms. We can obtain better results by using larger λ for ClusterM.

Table 5.8 exhibits the detailed results of various algorithms corresponding to MIPS data set for all protein interaction networks. The Acc scores and MMRs of ClusterM are consistently higher than other methods. For the impact of the choice

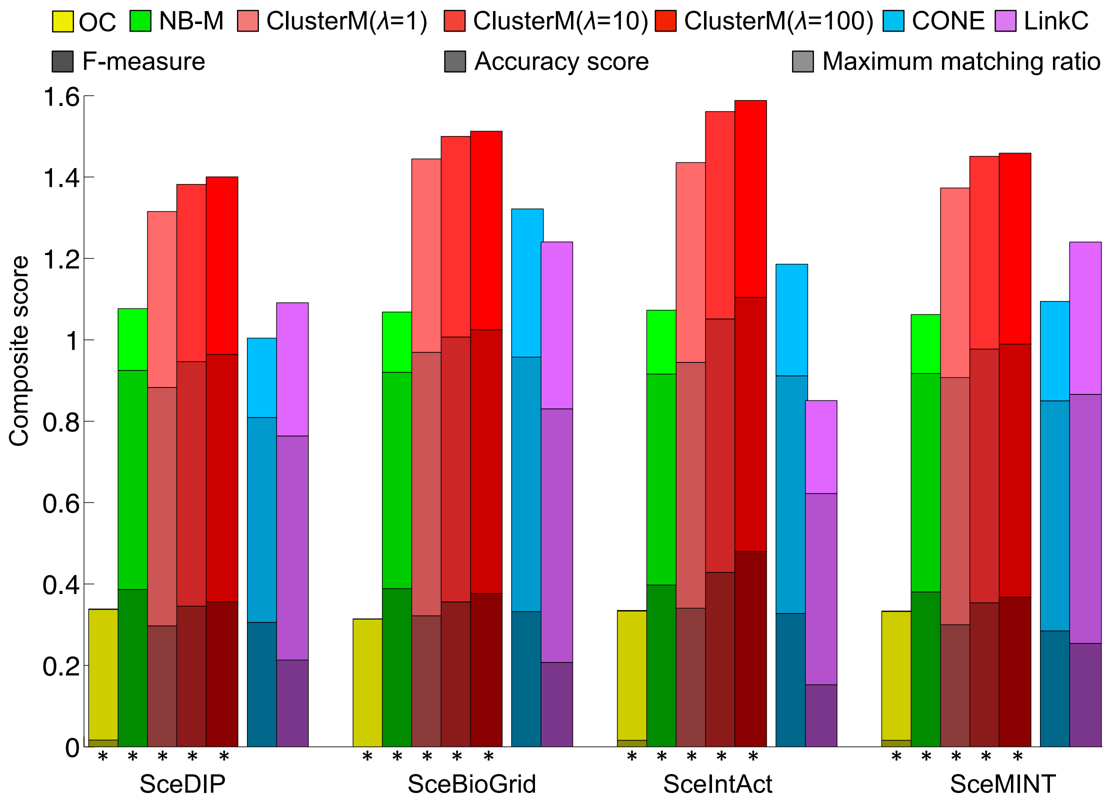


Figure 5.8: Results using SGD golden standard. Shades of the same color indicates quality scores of the same algorithm. The height of each bar is the value of the composite score. SceDIP, SceBioGrid, SceIntAct and SceMINT are four yeast protein interaction networks obtained from four different databases. Asterisks mark algorithms that use all four yeast protein interaction networks. CONE and LinkC are short for ClusterONE and LinkComm.

Table 5.7: Detailed benchmark results of various algorithms on four yeast protein interaction networks using the SGD complex set.

Dataset	Method	#. cluster	#. matched	F-m	Sn	PPV	Acc	MMR
SceDIP	OC	10	2	0.0164	**0.8703**	0.1184	0.3209	0.0013
	NB-M	486	91	**0.3864**	0.6416	0.4521	0.5386	0.1516
	CM($\lambda = 1$)	2885	**164**	0.2973	0.6596	0.5199	0.5856	0.4324
	CM($\lambda = 10$)	2660	160	0.3455	0.6762	**0.6538**	0.6008	0.4357
	CM($\lambda = 100$)	2510	157	0.3565	0.6715	0.5495	**0.6074**	**0.4365**
	CONE	380	86	0.3085	0.4082	0.6203	0.5032	0.1950
	LinkC	1838	137	0.2130	0.6290	0.4820	0.5506	0.3276
SceBioGrid	OC	10	0	0	0.8710	0.1133	0.3141	0
	NB-M	486	94	**0.3887**	0.6331	0.4464	0.5316	0.1479
	CM($\lambda = 1$)	3203	**180**	0.3219	0.7325	0.5723	0.6475	0.4752
	CM($\lambda = 10$)	2790	175	0.3516	0.7155	0.5918	**0.6507**	**0.4931**
	CM($\lambda = 100$)	2422	172	0.3765	0.6853	**0.6129**	0.6481	o.4881
	CONE	522	122	0.3318	0.6488	0.6035	0.6257	0.2642
	LinkC	5382	164	0.2074	**0.8880**	0.4373	0.6231	0.4100
SceIntAct	OC	10	2	0.0159	0.8773	0.1150	0.3176	0.0014
	NB-M	486	96	0.3974	0.6333	0.4244	0.5184	0.1569
	CM($\lambda = 1$)	2376	**175**	0.3406	0.6654	0.5485	0.6041	0.4909
	CM($\lambda = 10$)	1779	173	0.4286	0.6769	0.5730	0.6228	**0.5096**
	CM($\lambda = 100$)	1422	167	**0.4794**	0.6519	**0.5987**	**0.6247**	0.4841
	CONE	496	117	0.3275	0.5742	0.5944	0.5842	0.2742
	LinkC	1297	93	0.1525	**0.9223**	0.2393	0.4698	0.2285
SceMINT	OC	10	2	0.0160	**0.8709**	0.1148	0.3162	0.0013
	NB-M	486	89	**0.3806**	0.6427	0.4478	0.5365	0.1452
	CM($\lambda = 1$)	3045	**169**	0.3000	0.6939	0.5310	0.6070	0.4660
	CM($\lambda = 10$)	2895	166	0.3540	0.7062	0.5499	**0.6232**	**0.4737**
	CM($\lambda = 100$)	2756	163	0.3670	0.6939	0.5573	0.6219	0.4692
	CONE	513	110	0.2848	0.5370	**0.5954**	0.5654	0.2442
	LinkC	2201	144	0.2542	0.6757	0.5540	0.6119	0.3743

Abbreviations: OC = OrthoClust, CM = ClusterM, F-m = F-measure, CONE = ClusterONE, LinkC = LinkComm.

of λ, we observe similar patterns to its influence on SGD data set, which is larger λ makes F-measure higher and Acc score and MMR are not very sensitive to λ.

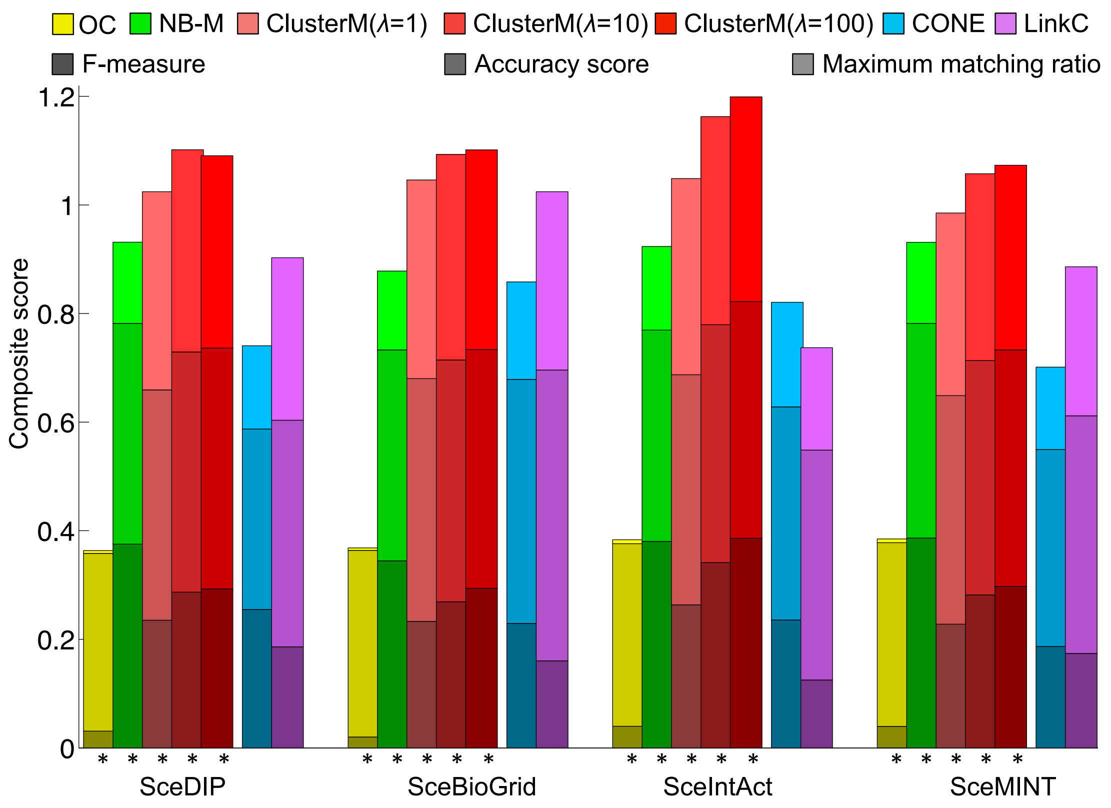


Figure 5.9: Results using MIPS golden standard. Shades of the same color indicates quality scores of the same algorithm. The height of each bar is the value of the composite score. SceDIP, SceBioGrid, SceIntAct and SceMINT are four yeast protein interaction networks obtained from four different databases. Asterisks mark algorithms that use all four yeast protein interaction networks. CONE and LinkC are short for ClusterONE and LinkComm.

Conserved module identification of yeast and human protein interaction networks

We further test the performance of various algorithms on identifying conserved modules across species. We detect conserved modules for pairwise networks, which are yeast protein interaction network SceDIP and human protein interaction network

Table 5.8: Detailed benchmark results of various algorithms on four yeast protein interaction networks using the MIPS complex set.

Dataset	Method	#. cluster	#. matched	F-m	Sn	PPV	Acc	MMR
SceDIP	OC	10	3	0.0309	**0.7444**	0.1440	0.3274	0.0051
	NB-M	486	74	**0.3753**	0.4587	0.3605	0.4067	0.1495
	CM($\lambda = 1$)	2885	**130**	0.2352	0.4479	0.4019	0.4243	0.3648
	CM($\lambda = 10$)	2660	127	0.2870	0.4653	0.4204	0.4423	**0.3725**
	CM($\lambda = 100$)	2510	120	0.2940	0.4611	**0.4291**	**0.4448**	0.3621
	CONE	380	74	0.2551	0.2749	0.4015	0.3322	0.1533
	LinkC	1838	109	0.1862	0.4775	0.3646	0.4173	0.2993
SceBioGrid	OC	10	2	0.0201	0.7589	0.1558	0.3438	0.0045
	NB-M	486	75	**0.3446**	0.4215	0.3577	0.3883	0.0045
	CM($\lambda = 1$)	3203	**131**	0.2332	0.4694	0.4255	**0.4470**	0.3658
	CM($\lambda = 10$)	2790	**131**	0.2692	0.4503	0.4401	0.4452	**0.3788**
	CM($\lambda = 100$)	2422	126	0.2941	0.4312	**0.4480**	0.4395	o.3678
	CONE	522	86	0.2293	0.4537	0.4452	0.4494	0.1795
	LinkC	5382	120	0.1604	**0.8179**	0.3504	0.5354	0.3285
SceIntAct	OC	10	4	0.0399	0.7549	0.1497	0.3362	0.0074
	NB-M	486	79	0.3802	0.4311	0.3514	0.3892	0.1540
	CM($\lambda = 1$)	2376	**125**	0.2636	0.4231	0.4245	0.4238	0.3612
	CM($\lambda = 10$)	1779	124	0.3414	0.4337	0.4423	**0.4380**	**0.3833**
	CM($\lambda = 100$)	1422	122	**0.3862**	0.4117	0.4611	0.4357	0.3772
	CONE	496	79	0.3256	0.3587	0.4296	0.3925	0.1927
	LinkC	1297	80	0.1251	**0.9028**	0.1986	0.4234	0.1880
SceMINT	OC	10	4	0.0397	**0.7541**	0.1519	0.3384	0.0068
	NB-M	486	79	**0.3866**	0.4330	0.3610	0.3954	0.1493
	CM($\lambda = 1$)	3045	**122**	0.2280	0.4334	0.4083	0.4207	0.3364
	CM($\lambda = 10$)	2895	119	0.2819	0.4468	0.4186	0.4315	**0.3441**
	CM($\lambda = 100$)	2756	115	0.2976	0.4343	**0.4366**	**0.4354**	0.3402
	CONE	513	67	0.1869	0.3274	0.4017	0.3626	0.1519
	LinkC	2201	100	0.1740	0.4744	0.4038	0.4377	0.2744

Abbreviations: OC = OrthoClust, CM = ClusterM, F-m = F-measure, CONE = ClusterONE, LinkC = LinkComm.

HsaDIP both obtained from DIP database. We compare our ClusterM with NetworkBlast (NB), MaWISh, NetworkBlast-M (NB-M) and OrthoClust (OC) on protein complex prediction based on reference protein complexes extracted from SGD, MIPS for yeast and CORUM for human. Additionally, ClusterM is also compared with ClusterONE (CONE) and LinkCommunity (LinkC).

Fig. 5.10 shows the comparison between the competing algorithm in terms of the composite score. Proteins between yeast and human protein interaction networks may not have many orthology relationships as two yeast protein interaction networks, therefore only part of the proteins are assigned to the conserved module, which makes the coverage of ClusterM, the number of proteins in the results yielded by ClusterM, smaller than CONE and LinkC, both of which take the whole network into account. For fair comparison, we only consider the modules generated by CONE and LinkC, where proteins are also covered by ClusterM. As shown in Fig. 5.10, comparing with other conserved module identification algorithms, ClusterM clearly has the best performance in each quality score on both SceDIP and HsaDIP networks over all these protein complex datasets, which reveals that ClusterM can recover more good-quality protein complexes of biological correspondence conserved in both yeast and human protein interaction networks. On the other hand, in comparison with CONE and LinkC, single network module identification algorithms, ClusterM also outperforms them over every protein complex dataset. The details of the quality scores, which are used in Fig.5.10, for all these protein complex datasets are listed in Table. 5.9, Table. 5.10 and Table. 5.11, respectively.

5.2.2.5 GO term consistency and coverage

We identify conserved modules for every combination (2-way, 3-way and 4-way) from protein interaction networks of four different species, which are SceDIP, HsaDIP,

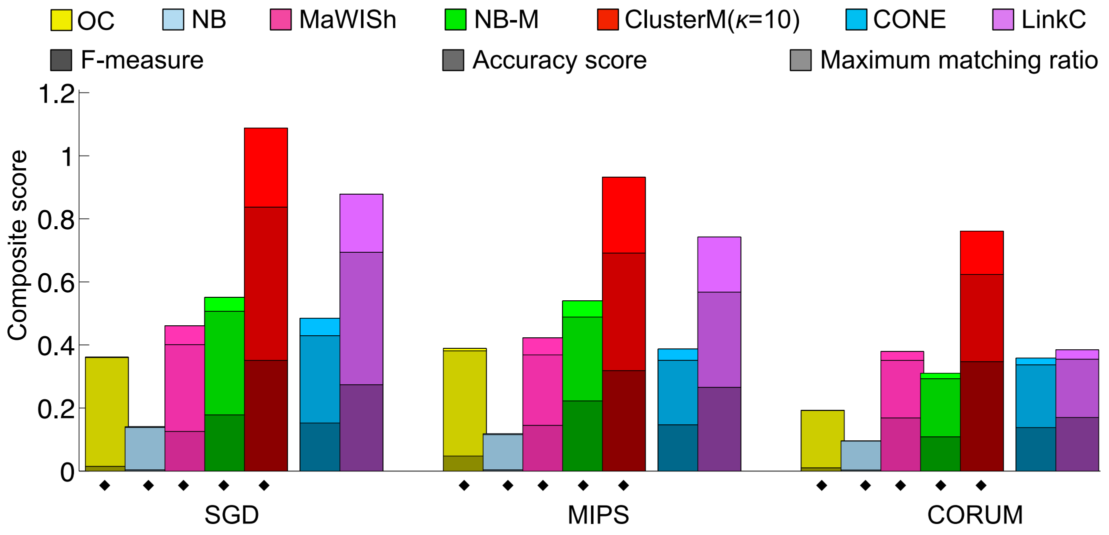


Figure 5.10: Comparison of composite scores for three protein complex dataset. Diamonds mark algorithms that use SceDIP and HsaDIP protein interaction networks. CONE and LinkC are short for ClusterONE and LinkComm.

Table 5.9: Detailed benchmark results of various algorithms on SceDIP and HsaDIP using the SGD complex set.

SGD	OrthoClust	NB	MaWISh	NB-M	CM($\lambda = 10$)	CONE	LinkC
Coverage	5081	336	427	656	2284	669	1088
#. modules	20	384	259	131	1018	128	535
#. matched	2	2	27	29	**110**	21	76
F-measure	0.0152	0.0040	0.1257	0.1784	**0.3513**	0.1524	0.2741
Sn	0.7227	0.0751	0.1636	0.3065	**0.4960**	0.1270	0.3145
PPV	0.1647	0.2404	0.4645	0.3527	0.4766	**0.6047**	0.5611
Acc	0.3450	0.1344	0.2756	0.3288	**0.4862**	0.2771	0.4201
MMR	0.0016	0.0025	0.0592	0.0438	**0.2506**	0.0552	0.1842

Abbreviations: CM = ClusterM, F-m = F-measure, CONE = ClusterONE, LinkC = LinkComm.

Table 5.10: Detailed benchmark results of various algorithms on SceDIP and HsaDIP using the MIPS complex set.

MIPS	OrthoClust	NB	MaWISh	NB-M	CM($\lambda = 1$)	CONE	LinkC
Coverage	5081	336	427	656	2284	669	1088
#. modules	20	384	259	131	1018	128	535
#. matched	5	2	28	32	**91**	19	64
F-measure	0.0478	0.0042	0.1453	0.2228	**0.3180**	0.1469	0.2657
Sn	0.6743	0.0638	0.1313	0.2439	**0.3396**	0.1046	0.2317
PPV	0.1720	0.1946	0.3790	0.2901	**0.4100**	0.4006	0.3939
Acc	0.3337	0.1114	0.2231	0.2660	**0.3731**	0.2047	0.3021
MMR	0.0076	0.0031	0.0542	0.0511	**0.2402**	0.0356	0.1749

Abbreviations: CM = ClusterM, F-m = F-measure, CONE = ClusterONE, LinkC = LinkComm.

Table 5.11: Detailed benchmark results of various algorithms on SceDIP and HsaDIP using the CORUM complex set.

CORUM	OrthoClust	NB	MaWISh	NB-M	CM($\lambda = 1$)	CONE	LinkC
Coverage	3827	416	522	639	1159	497	531
#. modules	20	382	182	131	699	121	449
#. matched	3	3	69	39	**233**	55	187
F-measure	0.0105	0.0036	0.1685	0.1089	**0.3472**	0.1382	0.1701
Sn	0.6667	0.0949	0.2219	0.3344	**0.4651**	0.1988	0.2112
PPV	0.0494	0.0883	0.1504	0.1018	0.1645	**0.1991**	0.1618
Acc	0.1815	0.0916	0.1827	0.1845	**0.2766**	0.1990	0.1849
MMR	0.0011	0.0008	0.0282	0.0171	**0.1373**	0.0212	0.0296

Abbreviations: CM = ClusterM, F-m = F-measure, CONE = ClusterONE, LinkC = LinkComm.

DmeDIP (fly protein interaction network obtained from DIP database) and CelDIP (worm protein interaction network obtained from DIP database). We annotate every protein in the aligned conserved modules, which contain proteins from every involved species, with their corresponding high-level GO terms and then compute the mean normalized entropy (MNE) over all aligned conserved module as our evaluation criterion for GO term consistency. Lower MNE value indicates more consistency of the GO terms shared by proteins in the aligned conserved module. From Table. 5.12, we observe that the aligned conserved modules detected by ClusterM have lower MNE values for most cases and ClusterM and OrthoClust achieve similar MNE scores. We then measure the coverage by the number of proteins in the aligned conserved modules and show the results in Table. 5.12. As shown in Table. 5.12, ClusterM has much better coverage than NB, MaWISh and NB-M. OrthoClust covers more proteins than ClusterM.

From Table. 5.12, we notice that aligned conserved modules identified by OrthoClust have low MNE values, which are competitive to ClusterM. Additionally, OrthoClust has better coverage than ClusterM. Therefore, we compare these two algorithms in more details in Table. 5.13. We define a aligned conserved module to be pure if at least half of the proteins in it annotated to one GO term. The purity of an aligned conserve module provides us another point of view to investigate the consistency of the GO terms. We also define a GO term is recovered if there exists an aligned conserved module, at least half of whose members are annotated by the GO term. The more recovered GO term we find, the more biological correspondence we can discover among different species. Table. 5.13 shows the comparison results on the basis of the average size of the aligned conserved modules, the number of pure aligned conserved modules and the number of recovered GO terms. From Table. 5.13, we observe that our ClusterM find more pure modules and more recovered GO terms, which means ClusterM can make more biological inference. OrthoClust although has the best coverage, the module sizes found by OrthoClust are too large to provide any specific biological insights. Therefore, ClusterM is better than OrthoClust in GO term consistency and biological inference.

5.2.2.6 Protein function prediction

In this section, we examine the capacity of protein function prediction based on guilt-by-association. Table. 5.14 illustrates the number of predictions each algorithm makes and the prediction precisions. We find that our ClusterM make the most number of predictions for all the selections of λ and achieve the best prediction precision except for pairwise alignment of HsaDIP and HasCel. MaWISh and NB-M attain the better precisions than ClusterM, however, MaWISh and NB-M only make 330 and 88 predictions, which means only the functions of 63 and 25 proteins are correctly predict. The precisions of ClusterM with different selections of λ are 20.5%, 20.2% and 22.5%, but functions of 258, 259 and 225 proteins are correctly predicted, which is much larger than MaWISh and NB-M.

Table 5.12: GO consistency and coverage comparison of all competing algorithms on DIP dataset.

Dataset	measure	NB	MaWISh	OC	NB-M	CM1	CM10	CM100
SH	MNE	6.186	5.471	4.898	5.997	4.131	4.145	**3.807**
	Coverage	744	986	**8779**	1349	5524	5224	3347
SD	MNE	3.809	2.772	2.380	3.486	2.162	2.142	**2.033**
	Coverage	747	1363	**12701**	2087	9167	8651	4358
SC	MNE	3.098	2.167	2.088	3.629	2.129	2.109	**1.948**
	Coverage	116	486	**7690**	651	4092	3782	1969
HD	MNE	7.639	6.009	4.226	7.030	4.146	4.171	**4.034**
	Coverage	735	1191	**11660**	1941	9123	8055	4190
HC	MNE	6.957	5.392	4.841	6.900	4.130	4.173	**3.919**
	Co.	49	561	6580	491	4256	3833	2051
DC	MNE	3.393	2.695	2.035	3.279	1.942	**1.885**	1.919
	Coverage	10	640	**10280**	1460	7006	6050	2104
SHD	MNE	—	—	3.768	6.095	4.130	4.144	**3.455**
	Coverage	—	—	**16717**	783	9247	8143	4999
SHC	MNE	—	—	3.894	6.324	4.022	3.975	**3.455**
	Coverage	—	—	**11660**	1004	5219	4704	2817
SDC	MNE	—	—	2.152	3.826	2.268	2.211	**1.858**
	Coverage	—	—	**15400**	1212	7759	6666	3748
HDC	MNE	—	—	3.682	5.753	3.962	3.951	**3.214**
	Coverage	—	—	**14406**	895	7784	6424	3629
SHDC	MNE	—	—	**3.182**	6.150	3.762	3.846	3.332
	Coverage	—	—	**19197**	1841	8255	6551	3378

Abbreviations: CM1 = ClusterM($\lambda = 1$), CM10 = ClusterM($\lambda = 10$), CM100 = ClusterM($\lambda = 100$) SH=SceDIP + HsaDIP, SD = SceDIP + DmeDIP, SC = SceDIP + CelDIP, HD=HsaDIP + DmeDIP, HC = HsaDIP + CelDIP, DC = DmeDIP + CelDIP, SHD = SceDIP + HsaDIP+DmeDIP, SHC = SceDIP + HsaDIP +CleDIP, SDC = SceDIP + DmeDIP + CelDIP, HDC = HsaDIP+DmeDIP+CelDIP, SHDC = SceDIP+HsaDIP+DmeDIP+CelDIP.

Table 5.13: GO consistency comparison between ClusterM and OrthoClust.

Dataset	method	Avg. size	#. pure	#. recovered GO
SH	OrthoClust	404.59	0	0
	ClusterM($\lambda = 1$)	9.06	664	447
	ClusterM($\lambda = 10$)	8.63	681	478
	ClusterM($\lambda = 100$)	6.45	685	687
SD	OrthoClust	706.33	0	0
	ClusterM($\lambda = 1$)	8.73	513	274
	ClusterM($\lambda = 10$)	8.08	568	325
	ClusterM($\lambda = 100$)	6.62	536	450
SC	OrthoClust	404.74	0	0
	ClusterM($\lambda = 1$)	8.36	198	158
	ClusterM($\lambda = 10$)	7.87	208	177
	ClusterM($\lambda = 100$)	6.34	208	279
HD	OrthoClust	532	1	2
	ClusterM($\lambda = 1$)	7.58	771	584
	ClusterM($\lambda = 10$)	6.69	863	713
	ClusterM($\lambda = 100$)	4.93	765	1023
HC	OrthoClust	235.29	2	8
	ClusterM($\lambda = 1$)	7.17	306	335
	ClusterM($\lambda = 10$)	6.68	339	400
	ClusterM($\lambda = 100$)	5.06	323	606
DC	OrthoClust	411.32	2	4
	ClusterM($\lambda = 1$)	7.23	96	87
	ClusterM($\lambda = 10$)	6.37	123	139
	ClusterM($\lambda = 100$)	4.83	128	162
SHD	OrthoClust	937.72	0	0
	ClusterM($\lambda = 1$)	14.09	474	192
	ClusterM($\lambda = 10$)	12.26	707	286
	ClusterM($\lambda = 100$)	8.15	1072	488
SHC	OrthoClust	588.9	0	0
	ClusterM($\lambda = 1$)	13.45	257	149
	ClusterM($\lambda = 10$)	11.93	355	189
	ClusterM($\lambda = 100$)	7.71	475	280
SDC	OrthoClust	811.68	0	0
	ClusterM($\lambda = 1$)	13.06	176	71
	ClusterM($\lambda = 10$)	11.39	243	135
	ClusterM($\lambda = 100$)	7.58	426	172
HDC	OrthoClust	688.71	0	0
	ClusterM($\lambda = 1$)	11.99	190	126
	ClusterM($\lambda = 10$)	9.89	306	210
	ClusterM($\lambda = 100$)	6.85	495	324
SHDC	OrthoClust	1076.89	0	0
	ClusterM($\lambda = 1$)	18.56	359	67
	ClusterM($\lambda = 10$)	14.89	593	142
	ClusterM($\lambda = 100$)	8.82	946	257

Abbreviations: SH=SceDIP + HsaDIP, SD = SceDIP + DmeDIP, SC = SceDIP + CelDIP, HD=HsaDIP + DmeDIP, HC = HsaDIP + CelDIP, DC = DmeDIP + CelDIP, SHD = SceDIP + HsaDIP+DmeDIP, SHC = SceDIP + HsaDIP +CleDIP, SDC = SceDIP + DmeDIP + CelDIP, HDC = HsaDIP+DmeDIP+CelDIP, SHDC = SceDIP+HsaDIP+DmeDIP+CelDIP.

Table 5.14: Comparison of protein function prediction for all competing algorithms.

Dataset	measure	NB	MaWISh	OC	NB-M	CM1	CM10	CM100
SH	#. predictions	221	831	0	530	**2457**	2443	1944
	precisions(%)	0	12.2	0	13.6	**19.5**	18.3	**19.5**
SD	#. predictions	99	886	0	605	2084	**2179**	1588
	precisions(%)	0	2	0	5.3	4.7	**6.3**	6.0
SC	#. predictions	44	309	0	181	**938**	937	756
	precisions(%)	0	1.3	0	0	6.6	7.2	**7.5**
HD	#. predictions	336	796	0	248	2714	**2885**	2088
	precisions(%)	0	11.9	0	4.8	14.9	14.8	**17.5**
HC	#. predictions	0	330	0	88	1260	**1281**	999
	precisions(%)	0	19	0	**28.4**	20.5	20.2	22.5
DC	#. predictions	0	209	0	49	386	**436**	371
	precisions(%)	0	0	0	0	2.1	1.6	**3.5**
SHD	#. predictions	—	—	0	783	2313	**2808**	2355
	precisions(%)	—	—	0	6.8	**10.3**	9.9	14.1
SHC	#. predictions	—	—	0	331	1435	**1567**	1257
	precisions(%)	—	—	0	0	**15.5**	14.6	14.7
SDC	#. predictions	—	—	0	230	916	**1082**	968
	precisions(%)	—	—	0	0	4.4	3.6	**5.8**
HDC	#. predictions	—	—	0	108	1031	**1264**	1203
	precisions(%)	—	—	0	0	10.1	7.4	**11.6**
SHDC	#. predictions	—	—	0	564	1369	**1836**	1538
	precisions(%)	—	—	0	0	13.44	11.0	**12.2**

Abbreviations: CM1 = ClusterM($\lambda = 1$), CM10 = ClusterM($\lambda = 10$), CM100 = ClusterM($\lambda = 100$) SH=SceDIP + HsaDIP, SD = SceDIP + DmeDIP, SC = SceDIP + CelDIP, HD=HsaDIP + DmeDIP, HC = HsaDIP + CelDIP, DC = DmeDIP + CelDIP, SHD = SceDIP + HsaDIP+DmeDIP, SHC = SceDIP + HsaDIP +CleDIP, SDC = SceDIP + DmeDIP + CelDIP, HDC = HsaDIP+DmeDIP+CelDIP, SHDC = SceDIP+HsaDIP+DmeDIP+CelDIP.

6. CONCLUSION

In this dissertation, we propose several algorithms for identifying functional modules for biological networks. We have developed algorithms in Chapters 3 and 4 to detect functional modules for individual and multiple networks based on the interaction patterns of the nodes. In Chapter 5, we propose an algorithm to deal with the degree heterogeneity for individual and multiple networks to find cohesive functional modules in biological networks. The mathematical framework and the developed algorithms used in this dissertation can be applied to any kind of networks not only biological networks. We summarize the important innovations from each of our contributions as follows:

6.1 Contributions for individual networks module identification

In the dissertation, we propose several module identification methods to detect functional modules with different topological structures. In chapter 3, we develope methods based on the concept of block modeling, which identifies modules based on the role that each node plays in the network. In chapter 5, we focus on finding cohesive modules, which are densely connected inside and loosely connected outside.

For module identification based on block modeling, we fist develop a sub-gradient algorithm with path generation heuristic to efficiently solve a classic block modeling framework using convex programming strategies. The algorithm provides biologists a useful tool to bird view the whole biological networks and have a better understand of the organization of the networks topologically. In order to look for small size functional modules, we develop SLCP2 based on tow-hop random walk on the networks. A spectral method is then used to find the low-conductance set in the transition matrix of the two hop random walk. Based on the framework of SLCP2, we propose an overlapping module identification algorithm, called GLCP2. We show that SLCP2 and GLCP2 can identify functional modules that other state-of-the-art algorithms can not find. Based on the non-negative matrix factorization framework, we propose a general framework which can handle both directed and indirected networks. Our APNMF algorithm has the convergence guarantee and can be efficiently solved by the proximal method. We showe that APNMF is the best NMF based algorithm for network clustering problem.

For protein complex prediction, we concentrate on the cohesive subnetworks in the networks. We devise an algorithm that can deal with the degree heterogeneity problem. Our FLCD can find cohesive modules with the present of nodes with low degrees. Protein complex prediction results show that FLCD is the best algorithm in detecting the protein complexes in protein interaction networks.

6.2 Contributions for multiple networks module identification

We propose algorithms that are the extension of the algorithms for individual networks. The algorithms we developed are scalable and easy to be parallelized. The superior performance has been illustrated by experiments.

The multiple network module identification algorithms based on block modeling introduced in section 4.1 can help us comprehensively detect the similar regions between two or more biological networks together. The algorithm also sets a good example to integrate different data sources (biological network and sequence). ASModel is the extension of SLCP2 for pairwise networks, which can find fine-size functional modules to relieve the resolution problems. ASModel is based on the two-hop random walk on the integration networks, which combines biological interactions and orthology relationships, and then uses the conductance concept to find meaningful modules exist in both networks.

We also propose a framework ClusterM to find conserved modules in multiple protein interaction networks based on FLCD. Our ClusterM is a local algorithm which is computational light and can easily handle multiple networks. Through experimental comparison, we conclude that ClusterM outperforms other developed algorithm on protein complex prediction and GO term consistency.

REFERENCES

[1] Y. Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[2] Daniel Aloise, Sonia Cafieri, Gilles Caporossi, Pierre Hansen, Sylvain Perron, and Leo Liberti. Column generation algorithms for exact modularity maximization in networks. *Phys Rev E*, 82:046112, 2010.

[3] SF Altschul, W Gish, W Miller, EW Myers, and DJ Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, 1990.

[4] Oscar Alzate. *Neuroproteomics*. CRC Press, 2009.

[5] L.N.F. Ana and A. K. Jain. Robust data clustering. In *CVPR*, 2003.

[6] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486, 2006.

[7] M Ashburner, CA Ball, JA Blake, and et al. Gene ontology: Tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, 2000.

[8] F. Bash, R. Jenatton, and J. Mairal G. Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[9] Sebastian Bauer, Steffen Grossmann, Martin Vingron, and Peter N. Robinson. Ontologizer 2.0—a multifunctional tool for go term enrichment analysis and data exploration. *Bioinformatics*, 24(14):1650–1651, 2008.

[10] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. on Imaging Sciences*, 2(1):183–202, 2009.

[11] DP Bertsekas. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999.

[12] H Bisgin and H.N. Dalfes. Parallel clustering algorithms with application to climatology. In *European Geosciences Union General Assembly*, 2008.

[13] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10:P10008, 2008.

[14] J.M. Borwein and A.S. Lewis. *Convex analysis and nonlin- ear optimization: theory and examples.* Springer, 2006.

[15] E Boyle, I Elizabeth, S Weng, J Gollub, H Jin, D Botstein, J.M Cherry, and G Sherlock. GO::TermFinder—open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes. *Bioinformatics*, 20:3710–3715, 2004.

[16] U Brandes, D Delling, M Gaertler, R Görke, M Hoefer, Z Nikoloski, and D Wagner. On modularity clustering. *IEEE Trans. on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[17] BJ Breitkreutz, C Stark, et al. The BioGRID Interaction Database: 2008 update. *Nucleic Acids Res*, 36:D637–640, Jan 2008.

[18] J. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences*, 101(12):4164–4169, 2004.

[19] J. Chan, W. Liu, A. Kan, and et al. Discovering latent blockmodels in sparse and noisy graphs using non-negative matrix factorisation. In *ACM International Conference on Information and Knowledge Management*, 2013.

[20] K. Chaudhuri, F. Chung, and A. Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition mode. *Journal of Machine Learning Research*, 2012:1–23, 2012.

[21] P.Y. Chen, C.M. Deane, and G. Reinert. Predicting and validating protein in- teractions using network structure. *PLoS Comput Biology*, 4(7):e1000118, 2008.

[22] G. Ciriello, M. Mina, P. H. Guzzi, M. Cannataro, and C. Guerra. Alignnemo: A local network alignment method to integrate homology and topology. *PLoS ONE*, 7(6):e38107, 2012.

[23] L Danon, J Duch, A Diaz-Guilera, and A Arenas. Comparing community structure identification. *J Stat Mech*, P09008, 2005.

[24] C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SIAM International Conference on Data Mining*, 2005.

[25] M. T. DIttrich, G. W. Klau, A. Rosenwald, T. Dandekar, and T. Muller. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231, 2007.

[26] S. Van Dongen. A cluster algorithm for graphs. *Technical Report INS-R0010*, 2000.

[27] R Drysdale. FlyBase : A database for the Drosophila research community. *Methods Mol Biol*, 420:45–59, 2008.

[28] N Fan and P. Pardalos. Multi-way clustering and biclustering by the ratio cut and normalized cut in graphs. *Journal of Combinatorial Optimization*, 23(2):224–251, 2010.

[29] N. Fan and P. Pardalos. Multi-way clustering and biclustering by the ratio cut and normalized cut in graphs. *Journal of Combinatorial Optimization*, 23(2):224–251, 2010.

[30] S Fortunato and M Barthélemy. Resolution limit in community detection. *Proc Natl Acad Sci USA*, 104(1):36–41, Jan 2007.

[31] AC Gavin, P Aloy, P Grandi, et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440:631–636, 2006.

[32] Mark B. Gerstein, Joel Rozowsky, Koon-Kiu Yan, and et al. Comparative analysis of the transcriptome across distant species. *Nature*, 512:445–448, 2014.

[33] R Guimeỳa and LAN Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.

[34] Inc. Gurobi Optimization. Gurobi optimizer reference manual.

[35] J. Hofman and C. Wiggins. A bayesian approach to network modularity. *Phys. Rev. Lett.*, 100:258701, 2008.

[36] P.W. Holland and S. Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[37] EL Hong et al. Gene ontology annotations at sgd: New data sources and annotation methods. *Nucleic Acids Res*, 36:D577–581, 2008.

[38] Young K. Yeast two-hybrid: so many interactions, (in) so little time. *Biol Reprod*, 58(2):302–311, 1998.

[39] M. Kalaev, V. Bafna, and R. Sharan. Fast and accurate alignment of multiple protein networks. *Journal of Computational Biology*, 16(8):989–999, 2009.

[40] BP Kelley, R Sharan, RM Karp, T Sittler, DE Root, BR Stockwell, and T Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proc Natl Acad Sci USA*, 100(20):11394–11399, 2003.

[41] S. Kerrien, B. Aranda, L. Breuza, and et al. The intact molecular interaction database in 2012. *Nucleic Acids Research*, 40(D1):D841–D846, 2012.

[42] S Kikugawa, K Nishikata, et al. Pcdq: human protein complex database with quality index which summarizes different levels of evidences of protein complexes predicted from h-invitational protein-protein interactions integra-

tive dataset. *BMC Systems Biology*, 6(Suppl 2)(S7), 2012.

[43] J. King. Conductance and rapidly mixing markov chains. Technical report, University of Waterloo, 2003.

[44] S Kirkpatrick, CD Gelatt Jr., and M Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[45] M Koyut*ü*rk, A Grama, and W Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *Bioinformatics*, 20:SI200–207, 2004.

[46] M. Koyuturk, A. Grama, and W. Szpankowski. Pairwise local alignment of protein interaction networks guided by models of evolution. In *RECOMB 2005*, pages 48–65, 2005.

[47] D Kuang, C. Ding, and H. Park. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the SIAM International Conference on Data Mining*, 2012.

[48] Licata L., Briganti L., Peluso D., and et al. Mint, the molecular interaction database: 2012 update. *Nucleic Acids Research*, 40:D857–D861, 2012.

[49] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys Rev E*, 80(016118), 2009.

[50] A. Lancichinetti, S. Fortunato, and K. Jnos. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[51] B Samuel Lattimore, Stijn Van Dongen, and M James C Crabbe. Genemcl in microarray analysis. *Computational Biology and Chemistry*, 29(5):354–359, 2005.

[52] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(788-791), 1999.

[53] X. Li, M. Wu, C. Kwoh, and S. Ng. Computational approaches for detecting protein complexes from protein interaction networks: a survey. *BMC Genomics*, 11(Suppl 1):S3, 2010.

[54] CS Liao, K Lu, M Baym, R Singh, and B Berger. IsoRankN: Spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25:253–258, 2009.

[55] Dirk M Lorenz, Alice Jeng, and Michael W Deem. The emergence of modularity in biological systems. *Phys Life Rev*, 8(2):129–160, 2011.

[56] Kaiser M. Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks. *New Journal of Physics*, 10(083942), 2008.

[57] Petersen M, Pardali E, and et al. Smad2 and smad3 have opposing roles in breast cancer bone metastasis by differentially affecting tumor angiogenesis. *Oncogene*, 29(9):1351–1361, 2010.

[58] J. Mairal. Optimization with first-order surrogate functions. In *International Conference on Machine Learning (ICML)*, 2013.

[59] D. Marbach, J. C. Costello, R. Kuffner, and et al. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9:796–804, 2012.

[60] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296:910–913, 2002.

[61] G.R. Mateus, M.G.C Resende, and R.M.A Silva. GRASP with path-relinking for the generalized quadratic assignment problem. *Journal of Heuristics*, pages 1–39, 2010.

[62] M. McDowall, M. Scott, and G. Barton. Pips: Human protein-protein interac-

tions prediction database. *Nucleic Acids Res*, 37:651–656, 2009.

[63] H.W. Mewes et al. Mips: Analysis and annotation of proteins from whole genomes. *Nucl. Acids Res.*, 32(D41-44), 2004.

[64] J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert. A lock-and-key model for protein-protein interactions. *Bioinformatics*, 22(16)(2012-2019), 2006.

[65] Jose C. Nacher and Jean-Marc Schwartz. Modularity in protein complex and drug interactions reveals new polypharmacological properties. *PloS ONE*, 7(1):e30028, 2012.

[66] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. *Processing of the 33rd International Conference on Management of Data*, pages 419–432, 2008.

[67] S. Navlakha, M.C. Schatz, and C. Kingsford. Revealing biological modules via graph summarization. *J. Comp. Biol.*, 16(2):253–264, 2009.

[68] T. Nepusz, H. Yu, and A. Paccanaro. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9:471–472, 2012.

[69] T Nepusz, H Yu, and A Paccanaro. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9(471-472), 2012.

[70] Y. Nesterov. *Introductory lectures on convex optimization.* Kluwer Academic Publishers, 2004.

[71] M Newman and E Leicht. Mixture models and exploratory data analysis in networks. *Proc Natl Acac Sci USA*, 204:9564–9569, 2007.

[72] M Newman and E Leicht. Mixture models and exploratory data analysis in networks. *Proc Natl Acac Sci USA*, 104:9564–9569, 2007.

[73] M. E. J. Newman. *Networks: An Introduction.* Oxford Univ. Press, 2010.

[74] MEJ Newman. Finding community structure in networks using the eigenvec-

tors of matrices. *Phys Rev E*, 74:036104, 2006.

[75] MEJ Newman and M Girvan. Finding and evaluating community structure in networks. *Phys Rev E*, 69:026113, 2004.

[76] N. P. Nguyen and M. T. Thai. Finding overlapped communities in online social networks with nonnegative matrix factorization. In *The 31st Military Communication Conference*, 2012.

[77] WR Pearson and DJ Lipman. Improved tools for biological sequence comparison. *Proc Natl Acad Sci USA*, 85(8):2444–2448, 1988.

[78] D. Pe'er and N. Hacohen. Principles and strategies for developing network models in cancer. *Cell*, 144(6):864–873, 2011.

[79] JB Pereira-Leal, AJ Enright, and CA Ouzounis. Detection of functional modules from protein interaction networks. *Proteins*, 54(1):49–57, Jan 2004.

[80] S Pinkert, J Schultz, and J Reichardt. Protein interaction networks: More than mere modules. *PLoS Comput Biol*, 6:e1000659, Jan 2010.

[81] C J Powers, S W McLeskey, and A Wellstein. Fibroblast growth factors, their receptors and signaling. *Endocrine-Related Cancer*, 7(165-197), 2002.

[82] TSK Prasad et al. Human Protein Reference Database—2009 update. *Nucleic Acids Research*, 37:D767–D772, 2009.

[83] J Reichardt. *Structure in Networks.* Springer-Verlag Berlin Heidelberg, 2008.

[84] J Reichardt. *Structure in Complex Networks.* Springer, 2009.

[85] J Reichardt and D.R. White. Role modules for complex networks. *Eur Phys J B*, 60:217–224, 2007.

[86] L. Royer, M. Reimann, et al. Unraveling protein networks with power graph analysis. *PLoS Comput Biol*, 4(7):e1000108, 2008.

[87] A. Ruepp, B. Brauner, I. Dunger-Kaltenbach, et al. Corum: The comprehensive resource of mammalian protein complexes. *Nucl. Acids Res.*, 36:D646–

D650, 2008.

[88] Fields S and Song O. A novel genetic system to detect protein-protein interactions. *Nature*, 340(6230):245–246, 1989.

[89] Sayed Mohammad Ebrahim Sahraeian and Byung-Jun Yoon. Smetana: Accurate and scalable algorithm for probabilistic alignment of large-scale biological networks. *PLoS ONE*, 8(7):e67995, 2013.

[90] L Salwinski, CS Miller, AJ Smith, FK Pettit, JU Bowie JU, and D Eisenberg. The Database of Interacting Proteins: 2004 update. *Nucleic Acids Research*, 32:D449–D451, 2004.

[91] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: Applications to community discovery. In *15th ACM SIGKDD international conference on knowledge discovery and data mining (KDD'09)*, 2009.

[92] V. Satuluri and S. Parthasarathy. Symmetrizations for clustering directed graphs. In *14th International Conference on Extending Database Technology (EDBT11)*, 2011.

[93] V. Satuluri, S. Parthasarathy, and D. Ucar. Markov clustering of protein interaction networks. In *ACM Conference on Bioinformatics, Computational Biology and Biomedicine 2010*, 2010.

[94] R. Sharan, S. Suthram, R.M. Kelley, and et al. Conserved patterns of protein interaction in multiple species. *Proc Natl Acac Sci USA*, 102:1974–1979, 2005.

[95] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[96] Y. Shih and S. Parthasarathy. Identifying functional modules in interaction networks through overlapping markov clustering. *Bioinformatics*, 28:i473–i479, 2012.

[97] Y. Shih and S. Parthasarathy. Scalable global alignment for multiple biological

networks. *BMC Bioinformatics*, 13(Suppl 3):S11, 2012.

[98] Yu-Keng Shih and Srinivasan Parthasarathy. Identifying functional modules in interaction networks through overlapping markov clustering. *Bioinformatics*, 28(18):1473–1479, 2012.

[99] R Singh, J Xu, and B Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proc Natl Acad Sci USA*, 105(35):12763–12768, 2008.

[100] Daniel Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC 2004*, 2004.

[101] C Stark, BJ Breitkreutz, T Reguly, L Boucher, A Breitkreutz, and M Tyers. BioGRID: A general repository for interaction datasets. *Nucleic Acids Res*, 34:D535–539, Jan 2006.

[102] Joshua M Stuart, Eran Segal, Daphne Koller, and Stuart K Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302(5643):249–255, 2003.

[103] Thomas C. J. Tan, Ruman Rahman, Farah Jaber-Hijazi, and et al. Telomere maintenance and telomerase activity are differentially regulated in asexual and sexual worms. *Proceedings of the National Academy of Sciences*, 109(11):4209–4214, 2012.

[104] R. Tatusov et al. The cog database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4:41, 2003.

[105] K. Voevodski, S. Teng, and Y. Xia. Finding local communities in protein networks. *BMC Bioinformatics*, 10(297), 2009.

[106] F. Wang, T. Li, X. Wang, and et al. Community discovery using nonnegative matrix factorization. *Data Min Knowl Disc*, 22(3):493–521, 2011.

[107] Y. Wang and X. Qian. Functional module identification by block modeling using simulated annealing with path relinking. In *ACM Conference on Bioinformatics, Computational Biology and Biomedicine 2012*, 2012.

[108] Y. Wang and X. Qian. A novel subgradient-based optimization algorithm for block model functional module identification. In *Asia Pacific Bioinformatics Conference 2013*, 2013.

[109] Y. Wang and X. Qian. A novel subgradient-based optimization algorithm for blockmodel functional module identification. *BMC Bioinformatics*, 14(Suppl 2):S23, 2013.

[110] Y Wang and X Qian. Functional module identification in protein interaction networks by interaction patterns. *Bioinformatics*, 30(1):81–93, 2014.

[111] Y. Wang and X. Qian. Functional module identification in protein interaction networks by interaction patterns. *Bioinformatics*, 30(1):81–93, 2014.

[112] Y. Wang and X. Qian. Joint clustering of protein interaction networks through markov random walk. *BMC Systems Biology*, 8(suppl 1):S9, 2014.

[113] Y Wang and X Qian. Joint clustering of protein interaction networks by block modeling. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1616–1620, 2014 © 2014 IEEE.

[114] Dunham WH, Mullin M, and Gingras AC. Affinity-purification coupled to mass spectrometry: basic principles and strategies. *Proteomics*, 12(10):1576–1590, 2012.

[115] E. Xing and M. Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical report, UC. Berkeley, 2003.

[116] Koon-Kiu Yan, Daifeng Wang, Joel Rozowsky, and et al. Orthoclust: an orthology-based network framework for clustering data across multiple species. *Genome Biology*, 15:R100, 2014.

[117] J Yang, X Liu, et al. Prevention of apoptosis by bcl-2: Release of cytochrome c from mitochondria blocked. *Science*, 275(1129-1132), 1997.

[118] H. Zha, C. Ding, et al. Spectral relaxation for k-means clustering. In *Neural Information Processing Systems 2001*, volume 14, pages 1057–1064, 2001.

[119] Shihua Zhang, Junfei Zhao, and Xiangsun Zhang. Common community structure in time-varying networks. *Phys Rev E*, 85(056110), 2012.

[120] Y. Zhang and D. Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2012.

[121] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. A greedy algorithm for aligning dna sequences. *J Comput Biol*, 7(1-2):203–214, 2000.