TOWARDS FINDING THE COMPLETE MODULOME: DENSITY CONSTRAINED BICLUSTERING

by

Recep Colak

B.Sc., Bilkent University, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

APPROVAL

Name:	Recep Colak
Degree:	Master of Science
Title of thesis:	Towards Finding The Complete Modulome: Density Constrained Biclustering

Examining Committee: Dr. Arthur (Ted) Kirkpatrick, Professor, Computer Science
Simon Fraser University
Chair

Dr. Martin Ester, Professor, Computer Science
Simon Fraser University
Senior Supervisor

Dr. Jian Pei, Professor, Computer Science
Simon Fraser University
Supervisor

Dr. Eldon Emberly, Professor, Physics
Simon Fraser University
Examiner

Date Approved: ____________________

Abstract

Large-scale gene expression experiments and interaction networks have become major data sources for discovery in systems biology. In several types of interaction networks, as is widely established, *active modules*, i.e. functional, simultaneously active groups of genes, are best encoded as highly interconnected regions that are co-expressed and show significant changes in an accompanying set of gene expression experiments. Accordingly, inferring an organism's *active modulome*, the entirety of active modules, translates to identifying these dense and co-expressed regions, which is $\mathcal{NP}$-hard.

We provide a novel algorithm, DCB-Miner, that addresses the corresponding computationally hard problem by means of a carefully designed search strategy, which has been specifically adapted to the topological peculiarities of protein interaction networks. Our algorithm outperforms all prior related approaches on standard datasets from *H. sapiens* and *S. cerevisiae* in a Gene Ontology-based competition and finds modules that convey particularly interesting novel biological meaning.

Keywords:
Systems biology; protein interaction networks; gene expression; dense subgraphs; biclustering

Subject Terms:
Data mining; Bioinformatics; Computational biology; DNA microarrays; Graph theory data processing

To my family

"I never guess. It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."

Sir Arthur Conan Doyle, 1859-1930

Acknowledgments

I would like to express my deepest gratitude to my senior supervisor, Dr. Martin Ester. I thank him for his continuous professional and personal support. He gave me motivation, support and patience to deal with all sorts of problems I have had during development of this thesis. I would like to thank my supervisor Dr. Jian Pei and my examiner Dr. Eldon Emberly for their constructive feedbacks at the initial stages of this thesis and also for reading and improving this thesis. I also thank Dr. Ted Kirkpatrick for chairing my thesis defense.

I would like to thank once more to Flavia Moser for the endless discussions we had, for her patience and for improving this thesis. I am very thankful to Dr. Jack Chen, Dr. Alexander Schönhuth and Jeffrey Shih-Chieh Chu for their help to improve the proposed model. My special thanks go to members of the SFU Data Mining Lab for the friendly environment. Finally, I also would like to thank to the CIHR/MSFHR Bioinformatics Training Program and its coordinator Sharon Ruschkowski and to Val Galat and Gerdi Synder of SFU School of Computing Science for the financial and administrative support.

This thesis would be impossible without the love and support of my dearest Pinar and my family. I want to thank you for the happiness and meaning you bring into my life.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

On the cellular level, life is driven by molecules acting in concert, in response to internal and external signals. The investigation of the inherent complex molecular patterns has been at the core of molecular biology since the discovery of the genetic code. The ultimate goal is to draw detailed maps of cellular mechanisms and their interplay. However, even in the post-genomic era, this is a hard task. Noisy experimental data and the superposition of many such mechanisms usually make direct computational approaches impossible. For this reason, a significant portion of the genes of the most studied model organisms lack comprehensive functional annotation. The situation is even worse in less studied organisms [66].

The modularity paradigm [4] facilitates to overcome these difficulties in a single step. In short, this systemic paradigm establishes that functional subunits of the cellular maps are encoded as *modules*. When mapped to the realm of biochemistry this translates to specific cellular functionality being explained by groups of genes rather than by single genes. Accordingly, in systems biology, a first worthwhile computational step is to identify an organism's modules. Provided with the *modulome* of an organism, that is, the entirety of its functional subunits, one can assign functions to not yet annotated gene products modularly associated with fully annotated functional partners. Furthermore, studying overlaps of modules and, in an even more general fashion, their hierarchical organization will finally help to draw more holistic pictures of an organism on the biochemical level.

Unlike other *omic* data types, which refer to large scale and holistic data gathered for understanding a specific aspect of the cellular life, there is no single experimental method or data source to be used to fully and directly annotate the modulome of an organism. As an example, consider the *genome* of an organism, which along with other genomes is usually a sufficient data source to find the complete set of genes of the organism to a substantial degree of correctness. Similarly, gene expression experiments are useful, if not sufficient, resources for constructing the transcriptome, i.e. the totality of expression profiles of genes under various conditions. Unfortunately, there is no such data source for construction of the modulome. Instead, the inference of modulome relies on analyzing and drawing conclusion from singleton (or a combination of other) omic data types. For example, finding co-expressed genes from transcriptome can yield insights into the modulome. Similarly, finding highly interacting groups of genes in the interactome usually translates to finding functional modules. As a final example, finding co-located clusters of genes along the genome often correspond finding to co-transcriptionally regulated genes, i.e. functional modules. All these translate to some form of non-trivial knowledge discovery from omic data types. This is why molecular biology has become one of the major application and innovation domains for data mining.

Various data mining methods, most significantly clustering and constrained pattern mining algorithms, have been developed to infer modules from single omic data types. As input, several biological omic data types are available, each describing a different aspect of the cellular system. Table 1 shows some of the popular and widely available omic data types in use today.

Data Type	**Type of Information**
Transcriptome	Expression of genes under various conditions
Interactome	Gene-protein and protein-protein interactions
Proteome	Activation of proteins under various conditions
Metabolome	Changes in concentrations of small molecules across conditions
Phenome	Gene-phenotype associations
Localizome	Cellular localization of individual proteins
Textome	Occurrence of genes/protein names in scientific articles

Table 1.1: Common *omic* data types

However, recently it became evident that joint analysis of multiple omic data types is much more promising than making inferences based on singleton data types [29]. Algorithms that integrate several types of datasets promise to be superior due to three important aspects:

1. High-throughput data of any type is still noisy to a substantial degree and/or incomplete. For example, high-throughput protein-protein interaction (PPI) identification experiments can contain up to 50% false positive interactions [81]. Similarly, noise in gene expression data is a prominent problem and has many origins.

2. Single data types only provide partial, highly specific information on the underlying biological system. For example, gene expression data only reflect cellular conditions on the transcriptional level, whereas interaction data only yield insight about particular features of more advanced cellular processes.

3. There is a global correlation between various omic data types [28, 30, 29]. Although some approaches used one data type to check the results from the analysis of another data type [76], it is sub-optimal. The global correlation between omic data types must be exploited by the learning algorithms for better performance.

Along these lines, this thesis tries to exploit the benefits of joint mining of multiple omic data types. We develop a novel way to efficiently combine transcriptome and interactome to find the *active modulome* of an organism. We refer to the entirety of the modules that are active with respect to the gene expression data employed as the *active modulome.* Supported by abundant scientific evidence, active modules are best identified as highly interconnected (*dense*) subnetworks where participating genes are co-expressed under specific biological conditions. We focus on transcriptome and interactome because (1) they are the most popular and widely available omic data types and (2) they serve as a basis for attribute based (phenome, localizome, epigenetic data, etc.) and graph based (interactome, protein structure data, gene association data, co-expression graphs, etc.) omic data types. Note finally that activity, as defined here, is relative to the gene expression experiments under consideration. As a consequence, completeness of the modulome is also relative to the cellular conditions explored.

1.1 Contributions

In this thesis, we present an algorithm that addresses prevalent difficulties for inferring modules by joint analysis of *transcriptome* and *interactome* data. As we shall see in the related work section, our algorithm integrates all desirable properties of existing module finding algorithms , which has not been done before. From the point of view of algorithmic complexity, the according search problem is $\mathcal{NP}$-hard. However, we demonstrate that, on the biological instances at hand, the problem becomes tractable by carefully designing the search strategy. We do this by combining the search for dense subnetworks with a biclustering method applied to as much as hundreds of different expression profiles.

In sum, our major contributions are:

- We *formally introduce the novel problem of finding density constrained biclusters (DCBs)*, i.e. densely connected subnetworks where nodes are subject to homogeneity constraints in a corresponding attribute space.
- We *design a novel search strategy*, Density Constrained Bicluster Miner (DCB-Miner) algorithm, which can efficiently solve this computationally hard problem for biological instances, namely PPI and genetic interactions (GI) networks on one hand and homogeneity constraints resulting from gene expression experiments on the other hand.
- We *use DCB-Miner to compute the active modulome of human and yeast* based on standard transcriptomic and interactomic datasets which has not been done before.
- We *demonstrate that our DCB-Miner algorithm clearly outperforms prior module finding approaches* in a Gene Ontology (GO) based evaluation procedure. This confirms the validity of the hypothesis of an active module being a dense and co-active subnetwork.
- We *show that DCBs can be used to computationally predict novel functional annotations* for crudely annotated and/or un-annotated genes.

1.2 Thesis Outline

The remainder of the thesis is organized as follows:

- In Chapter 2, we survey the related work.
- In Chapter 3, we formally introduce the Density Constrained Biclustering (DCB) problem, analyze its complexity and study the properties of the constraints a module must satisfy.
- In Chapter 4, we propose the DCB-Miner algorithm which exploits the properties derived in Chapter 3 and prove its correctness.

- In Chapter 5, we demonstrate the superiority of DCB-Miner algorithm in a Gene Ontology (GO) based competition against four state-of-the-art algorithms in use. We further present biological examples from yeast and human that convey particular interesting meaning. Finally, we report the results of run time experiments that show the scalability of DCB-Miner algorithm.
- We conclude with a summarization of our contributions and future extensions in Chapter 6.

Chapter 2

Related Work

In this chapter, we give a systematic analysis of the related work. From the computer science point of view, our work is closely related to clustering and constrained pattern mining. From a bioinformatics application point of view, our work corresponds to the functional module discovery from gene expression and protein interaction data.

The most widely used data mining methods for the functional module identification task are clustering and constraint based pattern mining. Clustering is defined as the process of grouping data objects into groups so that elements of a group are similar to each other and dissimilar to elements of other groups [31]. Unlike classification, class labels are not given apriori, therefore, it is an unsupervised learning process. Clustering has been widely studied within the data mining, statistics, pattern recognition, machine learning and bioinformatics communities. The type of the data (i.e. real-valued vectors, graphs, images, strings, etc.) to be clustered and the type of similarity metric usually depend on the domain of the data to be clustered. Constrained pattern mining, on the other hand, is the process of finding patterns in the data that satisfy some user defined constraints. Although the core task of pattern mining is the same as clustering, there exist certain differences between the two. First, patterns are typically smaller than clusters. Second, patterns are allowed to overlap, which might not be the case in most of the clustering algorithms. Finally, patterns are more interpretable than clusters due to their smaller size and to the fact that they satisfy user defined constraints.

Since both clustering and pattern mining are very broad topics, we will focus only on works that are developed or utilized for functional module discovery. In particular, we will give a special attention to biclustering and dense graph mining approaches as our algorithm combines ideas from both areas.

From this point on, we use the terms gene and protein interchangeably as a protein is directly identifiable from the gene coding for it.

2.1 Gene Expression Data (Transcriptome) Based Methods

Gene expression profiling is the process of measuring the activity of thousands of genes in a high-throughput manner in order to create a global picture of the cellular state. Usually, gene expression profiling data is represented as an $n \times m$ matrix E, containing expression levels for n genes (rows) under m experimental conditions (columns). Depending on the type of technology, the matrix entry $E_{i,j}$, $i \in 1..n$,$j \in 1..m$ can have different meanings. In cDNA microarray experiments, entry $E_{i,j}$ represents the (usually 2-logged) test vs. control fold change of gene i in condition j . For example, in Table 2.1, $E_{2,3} = -1.5$ means that expression value of gene B *decreased* $2^{1.5} = 2.82$-fold whereas $E_{5,3}$ indicates that expression value of gene B *increased* $2^4 = 16$-fold under Condition 3. On the other hand, in oligonucleotide array and SAGE (Serial Analysis of Gene Expression) based experiments, matrix entries represent the actual amount of gene expression. Hence there is no comparison, i.e. fold change, against the wild type.

Gene	**Cond-1**	**Cond-2**	**Cond-3**	**Cond-4**
A	2	0	-1	0
B	2	0.1	-1.5	0
C	0	0.1	0	0.2
D	1.1	2.1	0	2
E	-2	0	4	5

Table 2.1: Expression matrix from a cDNA experiment

Gene expression data has been the most widely used and publicly available attribute data for gene clustering and pattern mining. Therefore, the most classical post-genomic approach to module finding problem is to infer groups of co-expressed genes where expression patterns usually come from microarray experiments. As an example, consider again the toy data set given in Table 2.1, this time plotted in Figure 2.1 for easier interpretation. Gene A, B and E seem to be highly co-expressed. Note that, expression of genes A and B are positively correlated, whereas, E shows negative correlation with genes A and B. Moreover, D and C does not seem to be co-expressed with other genes. In summary, genes A, B and E may be involved in the same cellular process.

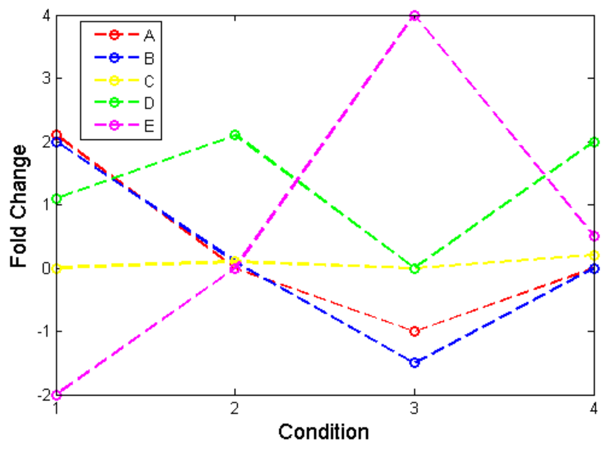


Figure 2.1: Plot of expression profiles of genes given in Table 2.1

A variety of classical approaches have demonstrated that, for a group of genes, co-expression significantly increases the likelihood of having similar function. In order to uncover such co-expression patterns, several clustering approaches including K-Means, Hierarchical Clustering and Self Organizing Maps (SOM) have been tested based on expression experiments to partition the set of genes into co-expressed groups [20, 74, 76]. Although these methods were quite useful, it is now well-known that some genes may be involved in different functional groups [27]. Therefore, a later generation of methods employed fuzzy clustering or mixture models to infer overlapping groups of genes [87, 27, 60] or more specifically address the challenges of clustering microarray data [34].

Classical clustering algorithms work in the full dimensional space, i.e. elements of a cluster are required to be similar in all dimensions. This may generate severe problems in the case of high dimensional data due to the so-called *curse of dimensionality*. For example, in the context of gene expression, members of a module may be co-expressed only under certain conditions and/or time points corresponding to a subspace of the full dimensional space. This is due to the complex gene regulation circuitry that results in temporal and spatial co-expression. Therefore, classical algorithms usually fail to capture subtle patterns that exist in subspaces. To deal with this problem, a novel class of methods called biclustering (or subspace clustering) algorithms [17, 75, 9, 91] were introduced. Due to the biological soundness, these methods have been found to perform significantly better than classical gene clustering algorithms. Basically, biclustering allows simultaneous clustering of rows (genes) and columns (experiments), meaning that the result is a set of genes with an associated subspace in which some type of *coherent behavior* is observed. Hence a bicluster induces a submatrix in the expression matrix. Similar to the classical clustering algorithms, the exact formulation of the problem and the coherency metric can lead to very different types of biclusters. The most widely used bicluster formulations are as follows [52]:

- *Bicluster with constant values*: The submatrix has the same values in all of the entries.
- *Bicluster with constant values on rows or columns*: The submatrix has constant expression values either along the rows or the columns.
- *Bicluster with coherent values*: Many definitions are proposed in this category. For example, some definitions may require the expression values of all genes to either decrease or increase simultaneously in each column. Alternative formulations may require all of the genes to induce the same linear ordering of experiments. Yet another definition of coherent behavior may be statistically defined. Therefore, this type of biclustering is more flexible and much closer to the reality.

Although not explicitly classified as biclustering algorithms, pattern mining algorithms have been utilized for the module identification problem as well. Frequent itemset mining [3, 32] approaches were used in [43, 8] to find small, overlapping set of genes (items) that are co-expressed (co-occur) in a large enough (i.e. satisfying a support threshold) subset of experiments (transactions). This was done by transforming the continuous gene expression matrix into a discrete one so that frequent itemset mining algorithms can be run on the data. This approach can be considered as another form of subspace clustering. Similarly, sequential patterns, which in this context are tuples consisting of a set of genes and a set of dimensions such that all the genes (item) induce the same linear ordering of experiments (transaction), were mined in [25].

Although gene expression data is the most publicly available omic data type, there is a lack of standardization in representation, storage and exchange of gene expression data. Moreover, data is produced in different research centers around the world that use various technological platforms and produce data at a variable level of quality. All these make the already hard problem of gene expression analysis even harder. Even the integration of these data sets poses a big challenge, let alone reliable cluster and pattern mining analysis. To address these problems, methods have been proposed which create a co-expression graph for each gene expression dataset, such that genes are the nodes and an undirected edge is inserted between a pair of genes if the genes satisfy a co-expression threshold (See Figure 2.2 for an example). It has been shown in [37, 84, 56] that dense subgraphs that frequently occur in multiple co-expression graphs correspond to functional modules. In these studies, co-expression networks of genes are constructed from up to 105 different human microarray datasets. Thereby, the module inference problem is transformed to that of finding dense subnetworks in the co-expression networks. In general, the higher the frequency of a dense subgraph, the higher the probability that the dense graph is a functional module. Combining various datasets in this way prevents over confident predictions based on low quality datasets.

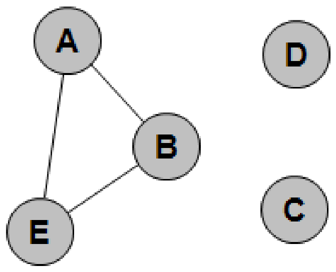


Figure 2.2: Co-expression graph extracted from the toy dataset given in Table 2.1 by applying an *Absolute Pearson Correlation Coefficient* threshold of 0.8. The highly co-expressed genes A,B and E induce a dense subgraph, in this case a clique of size 3, which may be a functional module. The higher the number of co-expression graphs in which A,B and E induce a clique, the higher the probability that A,B and E form a functional.

2.2 Interaction Data (Interactome) Based Methods

The second most widely used omic data type is the interactome, which is the whole set of molecular interactions in the cell. Depending on the type of interaction it can be represented as a directed, i.e. transcription factor - gene interactions, or as an undirected graph , i.e. protein-protein interactions and genetic interactions. Interaction networks have been used for identifying modules since they became available on a large-scale. Systematic analysis of interaction networks revealed many topological principles on the global organization [42, 14, 7, 58, 36]. Among these, the most important ones are *power-law* degree distribution, *small world effect*, *skewed graphlet distributions* and the *modular organization*, which is of particular interest for our work.

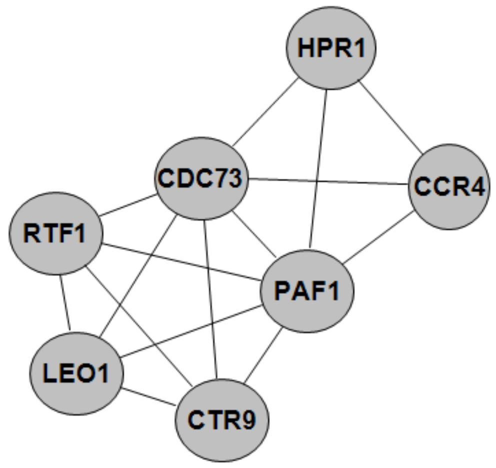


Figure 2.3: Example of a module which induces a dense subgraph in the interactome. Cdc73/Paf1 complex associates with RNA polymerase II and general RNA polymerase II transcription factor. It is involved in transcriptional initiation and elongation [11].

It is now well established that, in particular in protein-protein interaction (PPI) and genetic interaction (GI) networks, functional modules can be identified as dense subnetworks (See Figure 2.3 for an example) [78, 4, 92]. To infer modules from interaction networks alone, existing approaches mostly rely on PPI data. For example, [6] assigns weights to nodes to identify regions that are dense in terms of the weights. Others compute likelihood ratios of a subnetwork being a complex against occurring at random or employ various network-clustering algorithms [71, 46, 65, 90]. Many other network based prediction methods have been reviewed in a recent comprehensive study by [66] on standard datasets that have been proposed for evaluation and benchmarking competitions [12]. In this assessment, MCL, a Markov chain based method [21, 47] significantly outperformed the other ones. As a result of the underlying clustering techniques, these methods usually compute non-overlapping groups of proteins as modules. In a recent approach, to remove false positives detected by one method alone, modules are inferred by computing consensus clusters where clusters are obtained from several methods [5]. This can result in proteins being assigned to several clusters. Additional related approaches have been described in [66] and citations therein.

Dense graph mining has recently became a popular topic in data mining and many algorithms have been developed for addressing variants of the problem in various domains. Since these algorithms have been applied to or are applicable to biological networks, we believe it is useful to review them as well. The initial works in the data mining community focused on the *frequent subgraph* problem, i.e. finding subgraphs that frequently occur in at least a predefined number of input graphs . They usually accepted a collection of graphs and produced the frequent subgraphs that satisfy some *support* (frequency) constraint [41, 48]. Gradually, the focus moved to finding *frequent dense subgraphs* within a collection of graphs. In [82], an algorithm is proposed to find all frequent maximal cliques, i.e. complete graphs. The algorithm is based on a depth-first approach exploiting the anti-monotonicity of the *clique* property and the support properties, i.e. every subgraph of a frequent clique is also a frequent clique. A more relaxed density constraint requires only that the graph patterns are α-quasi-cliques, i.e. that every node has at least a specified percentage α, $0 < \alpha \leq 1$ of all possible edges within the pattern. [56] and [89] proposed new search space pruning strategies for efficiently mining all frequent, and all closed frequent resp., α-quasi-cliques. [85] also investigated the problem of mining all closed frequent graphs with edge connectivity at least k, where the edge connectivity is defined as the minimum cut size. The proposed CLOSECUT algorithm follows a pattern-growth approach and works well on datasets which contain mainly patterns with high support and low connectivity. The second algorithm, SPLAT, targets datasets containing mainly highly connected patterns. Relaxing the minimum support constraint, [37] presented an algorithm to mine subgraphs that are dense, defined based on the size of the minimum cut, and exhibit correlated occurrence across the collection of input graphs.

In the theory community, [1] proposed an approximation algorithm for finding the largest α-quasi-clique in disk resident data. It is based on a randomized greedy search procedure and is highly scalable to massive graphs. Although very efficient, it finds only the largest α-quasi-clique and is not very useful in biological applications considered here. Finally, graph partitioning algorithms such as normalized cut [68, 19] can be considered as another approach for finding some of the densest subgraphs. These algorithms partition the graph into components with small cut size, i.e. small weight of the edges between different components, which indirectly leads to dense components.

2.3 Methods That Jointly Mine Interactome and Transcriptome

Because of the availability of many large-scale datasets for an increasing number of organisms, a few rather sophisticated methods have been developed [54]. They usually augment gene expression data with metabolic [33], regulatory [63, 16, 61], PPI network [55], literature [86] and other diverse genomic information [45]. In [79], Bayesian network based framework was proposed for integrating diverse genomic data in order to find whether a given pair of genes are functional related or not. The approach is highly flexible, i.e. can integrate diverse data types and can assign weights on the reliability of the datasets based on prior knowledge. However, it can only predict pairwise relations between genes and does not explicitly draw conclusions at the module level. Related to our work are methods for the combined evaluation of PPI, GI and gene expression data. Among them, module finding approaches come closest to our work. [40] tries to find connected subnetworks which yield a high score measured in P-values obtained from gene expression experiments. In another classical approach by [33], a novel distance function, based on both expression and network information, are used in standard clustering procedures to partition the genes into groups. [62] employs more sophisticated statistical models to determine clusters of genes. In the most recent approach, [80] can test the hypothesis of a group of genes being co-expressed in a classical statistical procedure. Connected subnetworks which pass the test on a sufficiently high significance level are output as modules. Note that all of these methods yield non-overlapping modules as output. Density of the subnetworks is not addressed either. Some of these approaches restrict themselves to connected subnetworks [40, 80] while other address this requirement only implicitly [62, 33].

2.4 How Is Our Work Different ?

Based on the above discussions and *the wish list for clustering algorithms* [67], the following requirements should be taken into consideration while designing a useful and biologically meaningful module identification algorithm.

1. Algorithm should not require the number of clusters a priori, which itself is a challenging problem.

2. Due to high amount of noise, the data (or some part of it) may not really include a module. Hence, elements that are not related should not be forced to be in a cluster.

3. Most algorithms suffer from *the curse of dimensionality* problem. Therefore, the algorithm's performance should be robust to increasing dimensionality. Modules are usually active (co-expressed) only across a subset of conditions.

4. Modules should be overlapping because it is a known fact that many genes belong to multiple modules.

5. Algorithm should exploit the fact that elements of a module show correlation across various omic data types.

To our best knowledge, none of the methods mentioned above simultaneously address all of these criteria. By transforming these observations into mathematically well-defined constraints, we developed a constraint based pattern mining algorithm that can address all of the above mentioned challenges.

Chapter 3

Density Constrained Biclustering(DCB) Problem

In this chapter, we formally introduce the Density Constrained Biclustering (DCB) problem. and study its complexity. Moreover, we derive some useful properties of DCBs to be used in Chapter 4 for designing a search strategy for DCBs.

3.1 Problem Definition

We view the interaction network of an organism combined with fold-change gene expression data, i.e. cDNA microarray dataset, as an attributed graph.

Definition 1 (ATTRIBUTED GRAPH)**.** *An **attributed graph** is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, in which $\mathcal{V} = \{v_1, \ldots v_n\}$ denotes the node(gene) set, $\mathcal{E} \subseteq \{\{v_i, v_j\} \,|v_i, v_j \in \mathcal{V}, v_i \neq v_j\}$ denotes the edge (interaction) set and $\mathcal{A} : \mathcal{V} \rightarrow \mathcal{D}_1 \times \ldots \times \mathcal{D}_k$ is an attribute function, which assigns a k-dimensional attribute vector (expression profile) to each node $v \in V$. $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ is called the **attribute space** of $\mathcal{G}$ and $\mathcal{D}' \subseteq \mathcal{D}$ is called the **attribute subspace**.*

From the works on analysis of transcriptome and interactome surveyed in Chapter 2, we know that the following observations hold for functional modules:

1. Genes of the module are co-expressed in a subspace.

2. There exist a flow of information between elements of a module. Therefore, a module induces a connected subgraph in the interactome.

3. Members of a module are highly interconnected to each other while being loosely connected to elements of other modules.

For an induced attributed subgraph $G' = G[V'] = (V', E', \mathcal{A})$, we formalize the above facts as follows:

1. We say G' is **homogeneous** (co-expressed) in subspace D', $|D'| \geq \theta_{dim}$, if for all $d \in D'$:

$$|\max\{\mathcal{A}_d(v), v \in V'\} - \min\{\mathcal{A}_d(v), v \in V'\}| \leq \theta_h \tag{3.1}$$

   where $\mathcal{A}_d(v)$ denotes the attribute value of node v in dimension d and θ_h the homogeneity threshold. Informally, the homogeneity of G' requires that the attribute values (fold changes) of its nodes (genes) are within a range of at most θ_h across at least θ_{dim} dimensions. Moreover, if inequality 3.1 is not satisfied then G' is said no be **non-homogenous** on dimension d. Note that this definition of co-expression is a fairly restrictive one. For example, it does not capture negatively correlated regulation of genes. However, as we will elaborate in the following sections, it has very desirable computational properties that we will exploit in Section 3.3.

2. G' is **connected** if it is a connected component. This means that there exists a path between any pair of nodes in V', which consists only of nodes contained in V'.

3. The **density** of G', $d(G')$, is defined as G''s cliquishness, i.e. the ratio of the number of edges in G' over the number of possible edges in G',

$$d(G') = \frac{|E'|}{\binom{|V'|}{2}} = \frac{2|E'|}{|V'|(|V'| - 1)}. \tag{3.2}$$

   We say G' is α**-dense** if $d(G') \geq \alpha$, $0 \leq \alpha \leq 1$.

We call these constraints as *homogeneity*, *connectivity* and *density* constraints respectively. Now, we are ready to define a density constrained bicluster.

Definition 2 (DENSITY CONSTRAINED BICLUSTER (DCB))**.** *Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, homogeneity threshold θ_h, minimum dimensionality θ_{dim} and the density threshold α; an induced subnetwork $G' = G[V'] = (V', E', D', \mathcal{A})$ is called a **density constrained bicluster** (DCB) if*

- *G' is homogeneous wrt. θ_h and θ_{dim}*
- *G' is α-dense, i.e. $d(G') \geq \alpha$*
- *G' is connected.*

We say that a DCB satisfies the DCB constraint*.*

The number of density constrained biclusters can be prohibitive, therefore we restrict ourselves to only maximal ones, which is defined as follows.

Definition 3 (MAXIMAL DENSITY CONSTRAINED BICLUSTER)**.** *Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, a density constrained bicluster $G = G[V] = (V, E, D, \mathcal{A})$ is called a **maximal density constrained bicluster** , if $\nexists v \in \mathcal{V}$ and $\nexists D' \subseteq \mathcal{D}$ such that the graph $G' = (V \cup \{v\}, E', D', \mathcal{A})$ also satisfies the DCB constraint.*

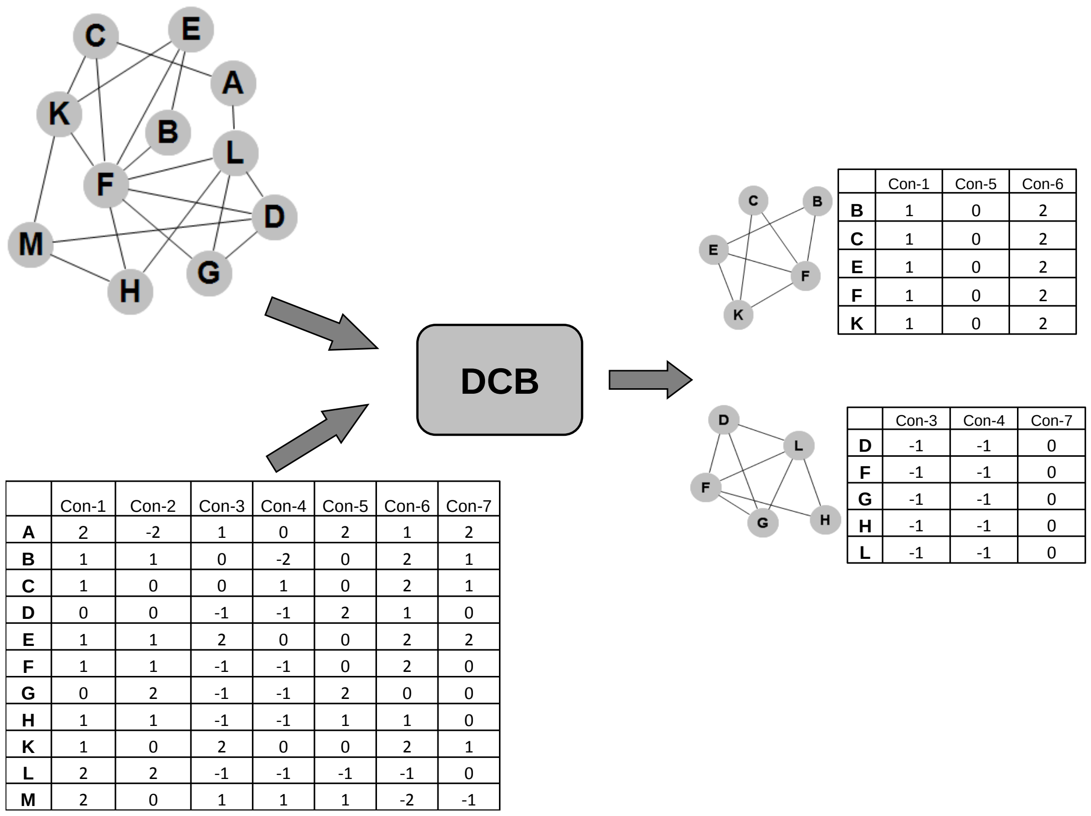
Translated back to the realm of biology, a *DCB* is a set of genes that are within a θ_h fold-change neighborhood of each other across at least θ_{dim} experimental conditions and whose associated nodes, which can usually be identified with their protein products, are densely interconnected in the interaction network (See Figure 3.1 for an illustration). According to our notation, we obtain the following computational problem:

Definition 4 (DENSITY CONSTRAINED BICLUSTERING (DCB) PROBLEM)**.** *Given an attributed graph, find all maximal DCBs.*
Input*: Attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, density threshold α, homogeneity threshold θ_h and minimum number of dimensions θ_{dim}.*
Output: *The set of all maximal DCBs specified by parameters α, θ_h and θ_{dim}.*

From this point on, we assume $\frac{1}{3} \leq \alpha \leq 1$. This is due to two reasons. First, density of known functional modules is usually much higher than $\frac{1}{3}$ (See Figure 5.1 and Section 5.2 for a detailed discussion). Second, α-dense graphs, $\alpha \geq \frac{1}{3}$ have amenable graph theoretic properties that makes efficient mining possible.

3.2 Complexity

Variants of dense subgraph search problems have been shown to be computationally hard problems. For example, finding the maximum clique in a graph is is known to be $\mathcal{NP}$-complete [44] and finding the complete set of $\alpha-$quasi-cliques, $0 < \alpha \leq 1$ is shown to be



	Con-1	Con-2	Con-3	Con-4	Con-5	Con-6	Con-7
A	2	-2	1	0	2	1	2
B	1	1	0	-2	0	2	1
C	1	0	0	1	0	2	1
D	0	0	-1	-1	2	1	0
E	1	1	2	0	0	2	2
F	1	1	-1	-1	0	2	0
G	0	2	-1	-1	2	0	0
H	1	1	-1	-1	1	1	0
K	1	0	2	0	0	2	1
L	2	2	-1	-1	-1	-1	0
M	2	0	1	1	1	-2	-1

	Con-1	Con-5	Con-6
B	1	0	2
C	1	0	2
E	1	0	2
F	1	0	2
K	1	0	2

	Con-3	Con-4	Con-7
D	-1	-1	0
F	-1	-1	0
G	-1	-1	0
H	-1	-1	0
L	-1	-1	0

Figure 3.1: Illustration of the DCB problem. For the given attributed graph the output contains two maximal DCBs wrt. parameters $\alpha = 0.7$, $\theta_{dim=3}$ and $\theta_h = 0$. For example, genes K,E,C,F and B form a highly connected module with the associated subspace consisting of conditions 1,5 and 6.

$\mathcal{NP}$-hard [56]. Moreover, [35] showed that even approximating the size of the maximum clique in polynomial time within a factor of n^{ϵ} ($\epsilon > 0$) is not possible unless $\mathcal{P} = \mathcal{NP}$. Since both cliques and α-quasi-cliques are α-dense graphs, it is not surprising that DCB problem is also a computationally hard problem.

Theorem 1 (COMPLEXITY)**.** *The DCB problem is $\mathcal{NP}$-hard.*

Proof. We do proof by restriction. Consider an instance of the DCB problem with parameters $\theta_h = \arg\max_{v_2,v_1,d} \{A_d(v_1) - A_d(v_2)\}, v_1, v_2 \in \mathcal{V}, d \in D$, $\theta_{dim} \geq 0$, $\alpha = 1$, i.e. attributes are omitted. Hence, the DCB problem includes as a special case the problem of finding the maximum clique, which is shown to be $\mathcal{NP}$-hard [56]. $\square$

Hence, the worst case instances of the DCB problem require an exhaustive enumeration of all 2^N subgraphs of $\mathcal{G}$, which is infeasible for realistic values of N, the number of nodes in $\mathcal{G}$. The average biological instance of the DCB problem, however, has certain properties that allow us to make the problem tractable by greatly reducing the search space. We use the DCB constraints of homogeneity, density and connectivity to prune the exponential search space that has to be explored. Therefore, we derive some useful properties of the DCB constraints in the rest of this chapter.

3.3 Properties of DCB Constraints

In this section, we analyze the properties of DCB constraints, which will serve as a search guidance for the DCB-Miner algorithm to be introduced in the next chapter. Like most of other pattern mining problems, DCB problem is also a computationally hard problem and therefore DCB-Miner relies on efficient search space prunning strategies based on the properties of the used constraints.

Recall that we defined an α-dense graph as a graph having at least α percentage of all possible edges. Next, we introduce two special types of α-dense graphs.

Definition 5 (α-QUASI-CLIQUE AND CLIQUE)**.** *Given a connected α-dense graph $G = (V, E)$. G is called **α-quasi clique** iff every node $v \in V$ has degree at least $\alpha(|V| - 1)$. A **clique** is a 1-quasi clique.*

We differentiate between the following types of nodes.

Definition 6 (α-REMOVABLE NODE, BRIDGE NODE, α-CRITICAL NODE and BRIDGE COMPONENT)**.** *Given a connected α-dense graph $G = (V, E)$ and a node $v \in V$.*

- *v is called **α-removable node** if $G - v$ is α-dense.*
- *v is called **bridge node** if $G - v$ is disconnected. Moreover, each of the connected components formed by removing non-bridge nodes from G is called a **bridge component**.*
- *G is called **α-critical**, if every α-removable node is a bridge node.*

We denote with $CN(G)$ the set of α-critical nodes and $B(G)$ the set of bridge nodes of graph G.

Definition 7 (α-CRITICAL GRAPH AND CRITICAL COMPONENT). *Given an α-dense graph $G = (V, E)$. G is called α-**critical**, iff G contains an α-critical node. The connected components in the subgraph induced by $CN(G)$ are called α-**critical components**.*

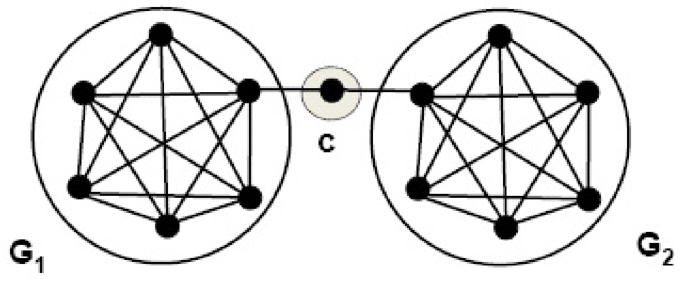


Figure 3.2: An example of an α-critical graph with the α-critical node c, $\alpha = 0.41$.

We illustrate some of the definitions in Figure 3.2. c is a bridge node because its removal disconnect the graph. Moreover, it is the only 0.41-removable node and is therefore a 0.41-critical node. Finally, the connected component consisting of node c, is the only 0.41-critical component.

Definition 8 (ISLAND COMPONENT and LEAF COMPONENT). *Let $G = (V, E)$ be a connected graph containing bridge nodes. The connected components induced by the nodes in $V \setminus B(G)$ are called **island components**. An island component which is connected to only one bridge node is called a **leaf component**.*

Looking at Figure 3.2 once again, we see that G_1 and G_2 are the only connected graphs upon removal of the set $CN(G) = \{c\}$ of critical nodes of G. Hence, G_1 and G_2 are **island components**. Since both of them are connected to only one α-critical node, namely node c, of only one critical component, they are also leaf components.

Definition 9 (α-STRONGLY CONNECTED GRAPH). *A graph $G = (V, E)$ is called α-**strongly connected** iff there exists at least one permutation $\tau = (v_{i_1}, \dots v_{i_n})$ over nodes in $V = \{v_1, \cdots v_n\}$ that induces a sequence $(G - \{v_{i_2}, \dots v_{i_n}\}, \dots, G - \{v_{i_{n-1}}, v_{i_n}\}, G - v_{i_n}, G)$, such that all graphs in the sequence are connected and α-dense.*

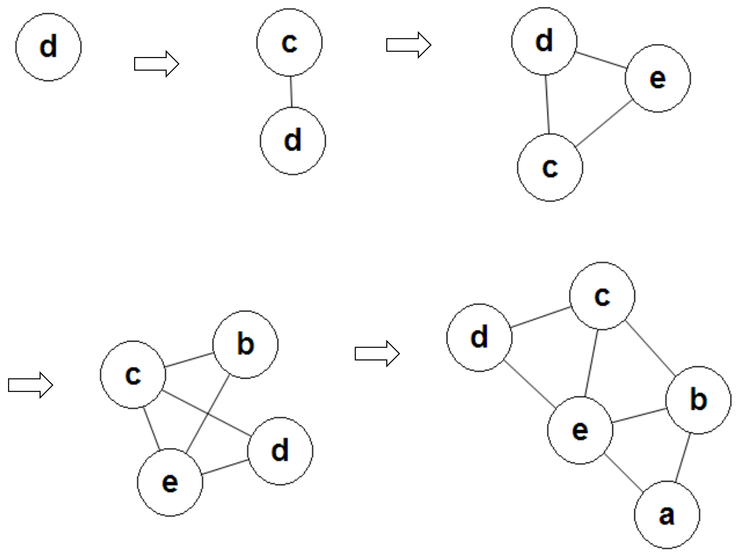


Figure 3.3: The node permutation $\{d, c, e, b, a\}$ induces connected subgraphs with densities $1, 1, 1, 0.83$ and 0.7 respectively. Therefore, the graph is α-strongly connected for $\alpha \leq 0.7$.

We illustrate the concept of α-strong connectivity in Figure 3.3. Note that the graph given in Figure 3.2 is not α-strongly connected for $\alpha = 0.41$. This is because any connected subgraph of size greater than 6 must include node c. Hence, any connected subgraph of size 12 contains c, which results in density smaller than 0.41. To sum up, the graph in Figure 3.2 cannot induce any permutations of type given in Definition 9. Indeed, if a graph is α-critical, then it cannot be α-strongly connected.

Finally, we conclude the definitions section with an interesting type of graph.

Definition 10 (MAXIMALLY EXPANDED-BY-ONE GRAPH). *Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, a DCB $G' = (V', E', D', \mathcal{A}) \subseteq G$ is called **maximally expanded-by-one** if either $G = G'$ or G' cannot be expanded by any neighboring node $v \in \mathcal{V} \setminus V'$ such that $G' + v$ is a DCB.*

In particular, we are interested in the anti-monotonicity, one of the most widely used constraints in pattern mining, properties of DCB constrains. Since, the characterization of constraints, i.e. anti-monotonicity and succinctness [53], of interest is essential for constraint based data mining, we start with defining types of constraints that are of interest in the context of the DCB problem.

Definition 11 (ANTI-MONOTONICITY)**.** *Given that a graph G satisfies a constraint C, C is called* ***anti-monotone*** *if all subgraphs of G satisfy C [53].*

For example, the constraint `G is a clique` is an anti-monotone constraint. This is because any subset of a clique is also a clique. It is trivial to show that simultaneous satisfaction of two anti-monotone constraint C_1, C_2 is an anti-monotone constraint $C_3 = C_1 \wedge C_2$. However, this is not valid for the following family of constraints.

Definition 12 (LOOSE ANTI-MONOTONICITY)**.** *Given that a graph $G = (V, E)$ satisfies C. C is called* ***loose anti-monotone*** *if there exists at least one subgraph of G with size $|V| - 1$ which fulfills C as well [10].*

Implicitly, the satisfaction of a loose anti-monotone constraint C requires existence of a permutation $\tau = (v_{i_1}, \dots v_{i_n})$ over nodes in $V = \{v_1, \cdots v_n\}$ that induces a sequence $(G - \{v_{i_2}, \dots v_{i_n}\}, \dots, G - \{v_{i_{n-1}}, v_{i_n}\}, G - v_{i_n}, G)$ of the nodes in V, such that each graph in this sequence satisfies C. Unlike anti-monotone constraints, simultaneous satisfaction of two loose anti-monotone constraints may or may not be a loose anti-monotone constraint. As we shall see later in this chapter, the difficulty in mining DCBs originate from the need for simultaneous satisfaction of two loose anti-monotone constraints, namely the connectivity and the α-density constraints. Finally, simultaneous satisfaction of a loose anti-monotone and an anti-monotone constraint is also a loose anti-monotone constraint.

Now that we have all the definitions we need, we start discussing the properties of α-density constraint. We start with properties of individual constraints, i.e. α-density, connectivity and homogeneity. Then, we will move to the properties of the DCB constraint , which requires simultaneous satisfaction of all the three individual constraints.

Theorem 2 (HOMOGENEITY)**.** *The homogeneity constraint is anti-monotone.*

Proof. Let a DCB $G = (V, E, D, \mathcal{A})$ satisfy the homogeneity constraint specified by the homogeneity threshold θ_h and minimum number of dimensions θ_{dim} in some subspace $D \subseteq \mathcal{D}$. By definition, this requires the following:

$$\forall d \in D : |\max\{A_d(v), v \in V\} - \min\{A_d(v), v \in V\}| \leq \theta_h \tag{3.3}$$

Note that reduction of the node set cannot increase the range in which the attributes fall, i.e. $\forall G' = (V', E', D', \mathcal{A}) \subseteq G$, we have $\max\{A_d(v), v \in \mathcal{V}\} \geq \max\{A_d(v), v \in \mathcal{V}'\}$. Similarly, $\min\{A_d(v), v \in \mathcal{V}\} \leq \min\{A_d(v), v \in \mathcal{V}'\}$. Therefore, we have:

$$\forall d \in D' : |\max\{A_d(v), v \in V'\} - \min\{A_d(v), v \in V'\}| \leq \theta_h \tag{3.4}$$

□

Next, we analyze graph related constraints and therefore omit the attributes.

Theorem 3 (α-DENSITY). *The α-density constraint is loose anti-monotone.*

Proof. Let $G = (V, E)$ be an α-dense graph. We need to show that there exists a node $v \in V$, such that $G' = (V', E') = G - v$ is also α-dense.

Since G is α-dense we know that $|E| = deg_G(v) + |E'| \geq \alpha\frac{|V|(|V|-1)}{2}$. We distinguish the following two cases:

- *There exists a node $v \in V$, such that $deg_G(v) < \lceil\alpha(|V|-1)\rceil$.* Then,

$$|E'| = |E| - deg_G(v) > \alpha\frac{(|V|-1)(|V|-2)}{2} = \alpha\frac{(|V'|)(|V'|-1)}{2} \tag{3.5}$$

  holds, which implies that G' is α-dense.

- *For every node $v \in V$, $deg_G(v) \geq \lceil\alpha(|V|-1)\rceil$.* In this case, we choose the node v with minimum degree k. Note that, $k \geq \lceil\alpha(|V|-1)\rceil$. Then, the density of G' is

$$d(G') \geq \frac{\frac{|V|k}{2} - k}{\frac{(|V|-1)(|V|-2)}{2}} \geq \frac{(|V|-2)k}{(|V|-1)(|V|-2)} \geq \frac{\lceil\alpha(|V|-1)\rceil}{(|V|-1)} \geq \alpha \tag{3.6}$$

  Therefore, G' is α-dense.

□

We now analyze connectivity, which is the last of the three individual DCB constraints.

Lemma 1. *Given a connected graph $G = (V, E)$, $|V| \geq 2$. There exist two distinct nodes $v_1, v_2 \in V$ such that both $G - v_1$ and $G - v_2$ are connected.*

Proof. We use induction on the number of nodes in G. If G does not contain any bridge node, then we are done. Otherwise, let v be a bridge node in G. Then $G - v$ consists of $l > 1$ connected components $G_1, \ldots G_l$. If G_1 has only one node, say u, then this node is not a bridge node in G hence $G - u$ is connected. Suppose G_1 has more than one node. By induction hypothesis, there are two distinct nodes u and w in G_1 such that they are not bridge nodes in G_1. We have the following cases:

- *Neither (v, u) nor (v, w) is an edge in E.* Then neither u nor w is a bridge node in G. The reason is as follows: Given that u, w are not bridge nodes in G_1, they can only be bridge nodes in G if they can disconnect G and $G \backslash G_1$, i.e. G_1 and v, which is not the case.

- *One of (v, u) and (v, w), say (v, u), is not an edge in E.* Then u is not a bridge node in G.

- *Both (v, u) and (v, w) are edges in E.* Then at least one of them, say u, is not a bridge node in G. The reason is as follows: $G_1 \backslash u$ $(G_1 \backslash w)$ is connected by definition and remains connected to $G \backslash G'$ via w (u).

In any case, we have at least one node in G_1 that is not a bridge in G. By symmetry, the same reasoning holds for G_2 as well and we have at least two non-bridge nodes in G. □

We can immediately conclude the following:

Theorem 4 (CONNECTIVITY)**.** *The connectivity constraint is loose anti-monotone.*

Proof. Let $G = (V, E)$ be a connected graph. We need to show that $\exists v \in V$ such that $G - v$ is connected. By Lemma 1, we have at least two such nodes. □

At this point we can summarize our findings from analysis of individual DCB constraints as follows:

- Simultaneous satisfaction of both the homogeneity and the connectivity or the homogeneity and the α-density constraints is a loose anti-monotone constraint.

- Simultaneous satisfaction of both the connectivity and the α-density is **not** a loose anti-monotone constraint. (See Figure 3.2 for an example).

We, therefore, conclude that DCB constraint is not anti-monotone and can at best be loose anti-monotone. As we shall see later in the text, under certain conditions, i.e. $\frac{1}{2} \leq \alpha \leq 1$, DCB constraint is loose anti-monotone. Moreover, we shall also observe that even though DCB constraint is not loose anti-monotone for $\frac{1}{3} \leq \alpha < \frac{1}{2}$, satisfaction of DCB constraint requires a restricted graph topology. Therefore, we need to analyze the cases $\frac{1}{2} \leq \alpha \leq 1$ and $\frac{1}{3} \leq \alpha < \frac{1}{2}$ separately. However, before doing so we make the following three observations, which hold for all connected graphs independent of α.

Lemma 2. *Let $G = (V, E)$ be a connected graph containing bridge nodes, i.e. $B(G) \neq \emptyset$. G contains at least two leaf components.*

Proof. We prove that there are two leaf components by induction on the number of bridge nodes in G. If there is only one bridge node, then we have at least two island components (See Figure 3.2 for an example). Suppose we have more than one bridge node. Let BC be one of the bridge components in G. Let $c_1, c_2 \in BC$ be two vertices, which are connected to the island components G_1 and G_2 respectively (See Figure 3.4 for an illustration). Note that c_1 and c_2 can be the same vertices, but G_1 and G_2 are distinct. If G_1 is connected to only one node in BC, namely c_1, then let L_1 be the subgraph induced by nodes in $V_1 \cup \{c_1\}$. Note that c_1 is not a bridge node wrt. the subgraph L_1. If G_1 is connected to a second node c_3, $c_1 \neq c_3 \in CC$, then there exists a connected component G_3 which is connected to only one node in BC, namely c_1 , otherwise c_1 would not be a bridge node . Let L_1 be the induced subgraph on $V(G_3) \cup \{c_1\}$. If L_1 is an island subgraph, then it is a leaf component. Otherwise by induction hypothesis over L_1, L_1 contains at least two leaf components such that at most one of them is connected to c_1. Thus, at least one of the other island components is a leaf component in G.

By symmetry, the same argument is applied to G_2 and we get another leaf component. Therefore, G contains at least two leaf components. □

Lemma 3. *Let $G = (V, E)$ be an α-critical graph and $v \in V$ be a node in a leaf component. Then, $deg_G(v) \geq \lceil \alpha(|V| - 1) \rceil$.*

Proof. Let $G_1 = (V_1, E_1)$ be a lead component and $v \in V_1$. Assume $deg(v) < \lceil \alpha(|V| - 1) \rceil$. Recall that in the first part of the proof of Theorem 3, we showed that every node with degree less than $\lceil \alpha(|V| - 1) \rceil$ is α-removable. We have two cases:

- *Removal of v disconnects G:* This means v is an α-critical node. However, by definition, leaf components cannot contain α-critical nodes. Hence, we have a contradiction.

- *Removal of v does not disconnect G:* This contradicts to the definition of leaf component - a leaf component cannot contain an α-removable nodes that does not disconnect G. Otherwise, G would not be α-critical.

Therefore, the degree of each node in an leaf component is at least $\lceil \alpha(|V| - 1) \rceil$. □

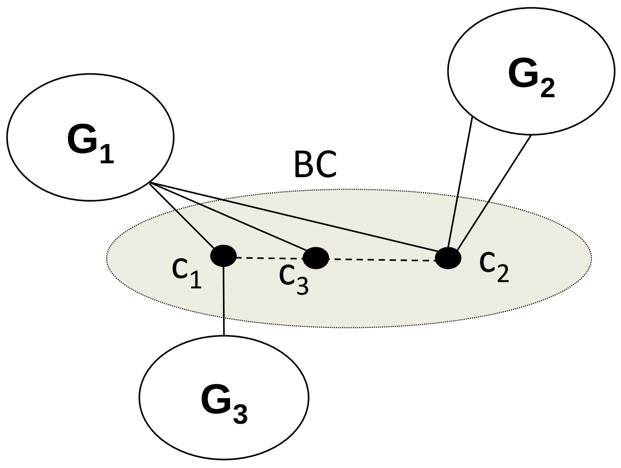


Figure 3.4: G_2 and G_3 are leaf components and BC is the bridge component. Although G_1 is also an island component, it is not a leaf component because it is connected to more than one bridge nodes, i.e. c_1,c_2 and c_3, of the bridge component BC.

Lemma 4. *Let $G = (V, E)$ be an α-critical graph. Not all nodes of a leaf component are connected to an α-critical node.*

Proof. Let $G_1 = (V_1, E_1)$ be a leaf component in G. Assume all nodes in G are connected to an α-critical node c. Then, trivially, $deg_G(c) \geq |V_1| + 1$. For all $v \in V$, the $deg_G(v)$ is bounded by $|V_1| - 1 + 1 = |V_1|$. This is because, v can be connected to all $|V_1| - 1$ neighbors in V_1 and at most one α-critical node, which is the node c in this case. Hence, $deg_G(v) < deg_G(c)$ holds. Note that, $G - c$ is α-dense by definition. Since $deg_G(v) < deg_G(c)$, $G - v$ must be α-dense as well. Moreover, by definition $G - v$ is connected. Hence, we have an α-removable node whose removal does not disconnect the graph. In this case, G cannot vbe α-critical. Hence, we have a contradiction. □

Intuitively, Lemma 4 implies that there is an upper bound on the degree of a critical node. If all nodes of a leaf component would be connected to an α-critical node, it would have such a high degree that it would not a be α-critical in the first place. Based on this, we can make the following observation.

Lemma 5. *Let $G = (V, E)$ be an α-critical graph. The size of a leaf component is greater than $\alpha(|V| - 1)$.*

Proof. Let $G_1 = (V_1, E_1)$ be a leaf component in G. We know by Lemma 4 that $\exists v \in V_1$ not connected to any α-critical node. Moreover, we also know by Lemma 3 that $deg_G(v) \geq \lceil \alpha(|V| - 1) \rceil$. Then, including v, the total number of nodes in the leaf component is at least $\lceil \alpha(|V| - 1) \rceil + 1 > \lceil \alpha(|V| - 1) \rceil > \alpha(|V| - 1)$. □

3.3.1 Properties of DCBs, $\frac{1}{2} \leq \alpha \leq 1$

Below is the summary of properties of connected α-dense graphs, $\frac{1}{2} \leq \alpha \leq 1$, that we will investigate in this section.

1. G does not contain an α-critical node (Lemma 6).
2. G is α-strongly connected (Lemma 7).

Lemma 6. *Let $\frac{1}{2} \leq \alpha \leq 1$ and $G = (V, E)$ be a connected α-dense graph. G does not contain an α-critical node.*

Proof. Assume G contains an α-critical node. By Lemma 2, there exist at least two leaf components G_1 and G_2 and at least one α-critical node. By Lemma 5, Both G_1 and G_2 contain more than $\alpha(|V| - 1) > \frac{|V|-1}{2}$ nodes. In total, G_1 and G_2 contain more than $2(\frac{|V|-1}{2}) = |V| - 1$ nodes. This is equivalent to G_1 and G_2 contain together at least $|V|$ nodes. Then, together with the α-critical node, $|V|$ contains at least $|V| + 1$, which leads to a contradiction. Therefore, G cannot contain an α-critical node. □

Lemma 7. *Let $\frac{1}{2} \leq \alpha \leq 1$ and $G = (V, E)$ be a connected α-dense graph. G is α-strongly connected.*

Proof. By Theorem 3, G has a non-empty set $R(G) \subseteq V$ of α-removable nodes. Moreover, by Lemma 6, $R(G)$ cannot contain any α-critical node. Therefore, we can find at least one non-bridge node $v \in R(G)$ such that $G - v$ is connected and α-dense. Applying this reasoning recursively, we can always find a connected α-dense subgraph with one node less at each step. Therefore, G is α-strongly connected. □

Theorem 5 ($\frac{1}{2}$ LOOSE ANTI-MONOTONICITY)**.** *DCB constraint is loose anti-monotone for $\frac{1}{2} \leq \alpha \leq 1$.*

Proof. By Theorem 2, $\forall v \in V$ $G - v$ is homogenous. Moreover, Lemma 7 requires $\exists v \in V$, $G-v$ is connected and α-dense. Hence, $G-v$ satisfies all three individual constraints of the DCB constraint. Applying this reasoning recursively, it is evident that DCB constraint is loose anti-monotone. □

3.3.2 Properties of DCBs, $\frac{1}{3} \leq \alpha < \frac{1}{2}$

In the following we analyze properties of connected α-dense graphs for $\frac{1}{3} \leq \alpha < \frac{1}{2}$. We will make use of the notation $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, which refer to the two of the possibly more (See Lemma 2) leaf components.

Below we summarize our major findings on properties of graphs α-dense ,$\frac{1}{3} \leq \alpha < \frac{1}{2}$, that we will investigate in this section.

1. G contains contains at most one α-critical component connecting two leaf components (Theorem 6).
2. No leaf component can contain more than twice as many nodes as the other leaf component (Lemma 8).
3. Both G_1 and G_2 are at least $\frac{1}{2}$-dense graphs (Lemma 9), hence they are at least $\frac{1}{2}$-strongly-connected (Corollary 1).
4. G can be decomposed into two overlapping α-strongly-connected subgraphs G_1' and G_2' respectively (Lemma 2).

Theorem 6 ($\frac{1}{3}$-RESTRICTED TOPOLOGY)**.** *Let $\frac{1}{3} \leq \alpha < \frac{1}{2}$ and $G = (V, E)$ be a connected α-dense graph. G contains at most two island components. Moreover, these island components are leaf components.*

Proof. We have the following cases:

- *G does not contain an α-critical node.* Then we are done.

- *G contains an α-critical node with two island components.* Then, by Lemma 2, there exist at least two leaf components G_1 and G_2. If G_1 and G_2 are the only leaf components, then we are done (See 3.5 for an example). Otherwise, there must be at least one more island component G_3. By Lemma 5, both G_1 and G_2, contain more than $\alpha(|V|-1) \geq \frac{|V|-1}{3}$ nodes. In total, G_1 and G_2 contain more than $2(\frac{|V|-1}{3})$ nodes. Thus, the size of the subgraph $G - G_1 - G_2$, which contains G_3, at least one α-critical node and possibly more nodes, is less than $|V| - (\frac{2|V|-1}{3}) = \frac{|V|+2}{3}$. However, by Lemma 3, $\forall v \in G_3$, v must be connected to more than $\frac{|V|-1}{3}$ nodes in $G - G_1 - G_2$, which requires the size of $G - G_1 - G_2$ be more than $\frac{|V|-1}{3} + 1 = \frac{|V|+2}{3}$. Thus, we have a contradiction and G cannot contain three or more island components. □

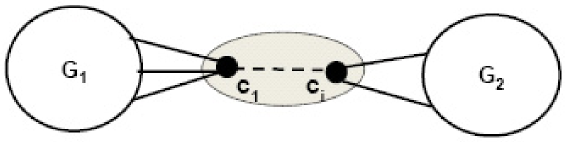


Figure 3.5: Illustration of the restricted graph topology for α-critical graphs, $\frac{1}{3} \leq \alpha < \frac{1}{2}$. The two leaf components ,G_1 and G_2 are connected via a simple path of i, $i \geq 1$, α-critical nodes $c_1, \ldots c_i$.

Unlike the case of $\frac{1}{2} \leq \alpha \leq 1$, we do not have the nice property of being loose anti-monotone in the case of $\frac{1}{3} \leq \alpha < \frac{1}{2}$. This is because for a connected α-dense graph $G = (V, E)$, the existence of an α-critical node means the absence of a node such that $G - v$ is both connected and α-dense. Therefore, simultaneous satisfaction of connectivity and α-density constraints is **not** loose anti-monotone. However, thanks to Theorem 6, we concluded that such a graph G must have a special topology, i.e. G contains two leaf components G_1 and G_2 connected via an α-critical component. Note also that, the α-critical component must be a simple path, i.e. it cannot contain a cycle. This is because if it contains one, then removal of one of the α-critical nodes, say v, would not make the $G - v$ disconnected, which contradicts with the fact that v is an α-critical node.

In the following, we will make further observations regarding the topology of α-critical graphs. These will be very useful when we discuss strategies for mining such graphs.

Lemma 8. *Let $\frac{1}{3} \leq \alpha < \frac{1}{2}$ and $G = (V, E)$ be an α-critical graph. Let also $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the two leaf components. Then, $\frac{1}{2} < \frac{|V_1|}{|V_2|} < 2$*

Proof. We do proof by contradiction. By Lemma 5, we have

$$\alpha(|V| - 1) < |V_1| \tag{3.7}$$

and

$$\alpha(|V| - 1) < |V_2| \tag{3.8}$$

Assume, $2|V_2| \leq |V_1|$. Using this assumption, (3.7) and (3.8) result in

$$2\alpha(|V| - 1) < |V_1|.$$

adding (3.8) gives us

$$3\alpha(|V| - 1) < |V_1| + |V_2|$$

Moreover, we know that $|V_1| + |V_2| = |V| - |CN(G)|$, therefore,

$$3\alpha(|V| - 1) < |V| - |CN(G)|$$

$$\alpha < \frac{|V| - |CN(G)|}{3(|V| - 1)}$$

Since $|CN(G)| \geq 1$, we get $\alpha < \frac{1}{3}$. This contradicts with $\alpha \geq \frac{1}{3}$. Therefore, $2|V_2| > |V_1|$. Using the same arguments, we also have $2|V_1| > |V_2|$. □

We are now ready to investigate the following useful observation on the density of a leaf component.

Lemma 9. *Let $\frac{1}{3} \leq \alpha < \frac{1}{2}$ and $G = (V, E)$ be a connected α-critical graph containing leaf components $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Both G_1 and G_2 are at least $\frac{1}{2}$-dense.*

Proof. Assume G_1 is not $\frac{1}{2}$-dense. This means there exists a node $v \in V_1$ which is connected to less than $\frac{|V_1|-1}{2}$ nodes within G_1 and to at most one critical node. Therefore, we have

$$deg(v) < \frac{|V_1| - 1}{2} + 1 = \frac{|V_1| + 1}{2} \tag{3.9}$$

Since v is not an α-critical node, we also know that $deg(v) > \alpha(|V| - 1)$.

Since $\alpha \geq \frac{1}{3}$ and $|CN(G)| \geq 1$,

$$deg(v) > \alpha(|V|-1) \geq \frac{1}{3}(|V_1|+|V_2|+|CN(G)|-1) \geq \frac{1}{3}(|V_1|+|V_2|) \tag{3.10}$$

Combining Equation 3.9 and 3.10, we have

$$\frac{1}{3}(|V_1|+|V_2|) < deg(v) < \frac{|V_1|+1}{2} \tag{3.11}$$

We distinguish between $|V_1|$ being odd and even.

- $|V_1|$ *is odd:* Since degree of v must be an integer,

$$deg(v) < \left\lceil \frac{|V_1|+1}{2} \right\rceil \Rightarrow deg(v) \leq \frac{|V_1|-1}{2} \tag{3.12}$$

  By Lemma 8, we have $\frac{|V_2|}{|V_1|} > \frac{1}{2}$. Since $|V_2|$ is an integer, we get

$$|V_2| \geq \frac{|V_1|+1}{2}.$$

  Combining this result with (3.11) we get

$$\begin{aligned} deg(v) &\geq \left\lceil \frac{1}{3}(|V_1|+|V_2|) \right\rceil \\ &\geq \left\lceil \frac{1}{3}\left(|V_1| + \frac{|V_1|+1}{2}\right) \right\rceil \\ &= \left\lceil \frac{|V_1|}{2} + \frac{1}{6} \right\rceil = \frac{|V_1|+1}{2} \end{aligned}$$

  (3.12) and (3.13) are contradicting each other, therefore G_1 is at least $\frac{1}{2}$-dense, if $|V_1|$ is odd.

- $|V_1|$ *is even:*

$$deg(v) < \frac{|V_1|+1}{2} \Rightarrow deg(v) \leq \frac{|V_1|}{2} \tag{3.13}$$

  Similarly, by Lemma 8, we have $\frac{|V_2|}{|V_1|} > \frac{1}{2}$. Thus,

$$|V_2| \geq \frac{|V_1|+2}{2}.$$

  From (3.13) we get,

$$\begin{aligned} deg(v) &\geq \left\lceil \frac{1}{3}(|V_1|+|V_2|) \right\rceil \\ &\geq \left\lceil \frac{1}{3}(|V_1| + \frac{|V_1|+2}{2}) \right\rceil = \frac{|V_1|+2}{2} \end{aligned} \tag{3.14}$$

  Since (3.13) and (3.14) contradict each other, G_1 is $\frac{1}{2}$-dense, if $|V_1|$ is even.

We conclude that in both cases G_1 is at least $\frac{1}{2}$-dense. By symmetry, we have a similar proof for G_2. □

Corollary 1. *Let $\frac{1}{3} \leq \alpha < \frac{1}{2}$ and $G = (V, E)$ be a connected α-critical graph containing leaf components $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Both G_1 and G_2 are at least $\frac{1}{2}$-strongly-connected.*

Proof. Immediately from Lemma 7 and Lemma 9. □

Although, we have not discussed the algorithmic implications loose anti-monotonicity, as we shall see in the next chapter, it is a desirable property. In this respect, leaf components, being at least $\frac{1}{2}$-strongly-connected, can be mined by a carefully designed algorithm. However, this is not enough as we need to find the maximal dense graph, which consists of two island components plus the critical component connecting these two leaf components. We now establish another nice property of leaf components, in which we show that we can find supergraphs of the two leaf components such that they are α-strongly-connected and they overlap.

We start with analyzing the density of supergraphs of leaf components that are formed by expanding the leaf components by their immediate critical bridge neighbors (See Figure 3.6).

Lemma 10. *Let G be an α-critical graph containing an α-critical component CC. Let also $c_1 \in CC$ and $c_2 \in CC$ be the α-critical nodes that G_1 and G_2 respectively are connected to. Let finally $G_1^* = G_1 + c_1$ and $G_2^* = G_2 + c_2$. G_1^* and G_2^* are both α-dense.*

Proof. For every node $v \in G_1$, we have $deg(v) > \alpha(|V| - 1)$ by Lemma 3. Therefore, we have

$$|E_1^*| > \frac{|V_1|\alpha(|V| - 1)}{2}$$

Calculating the density of G_1^*, we get

$$\begin{aligned} d(G_1^*) &= \frac{2|E_1^*|}{|V_1|(|V_1| + 1)} \\ &> \frac{2(|V_1|\alpha(|V| - 1))}{2(|V_1|(|V_1| + 1))} \\ &= \frac{\alpha(|V_1| + |V_2| + |CN(G)| - 1)}{|V_1| + 1} > \alpha \end{aligned} \tag{3.15}$$

Since $|V_2| + |CN(G)| - 1 > 1$, we conclude that G_1^* is α-dense. Analogous proof holds for G_2^*. □

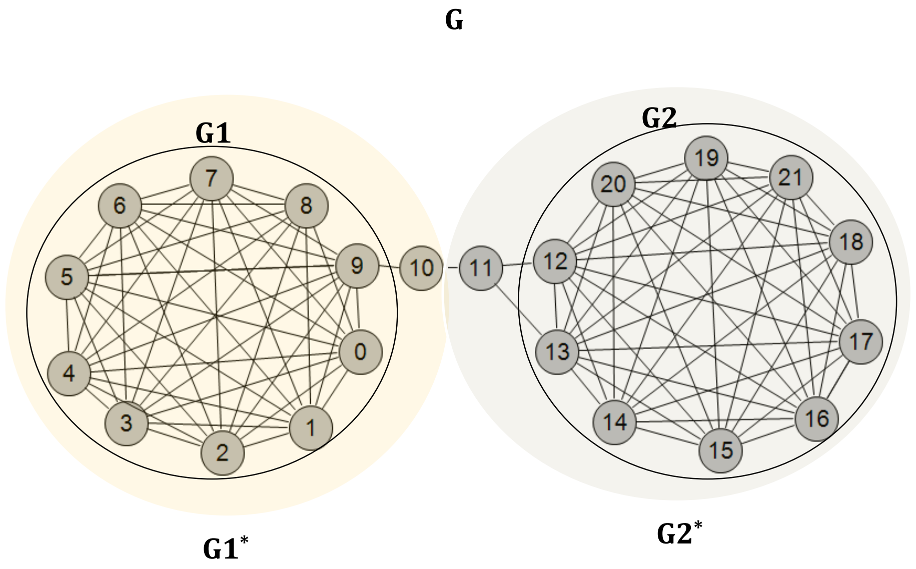


Figure 3.6: G is α-critical for $\alpha = 0.406$ and G_1 and G_2 are the leaf components. G_1^* and G_2^* are are both 0.406-dense.

Before analyzing properties of supergraphs of leaf components that include more nodes than just the immediate α-critical node neighbor, we make the following trivial observation.

Lemma 11. *The size of a graph which can contain an α-critical node is at least 9. The size of an island components is at least 4.*

Proof. The smallest graph contains only one α-critical component CC consisting of one node $c \in CC$, $deg(c) \geq 2$. In order for c to be α-critical, its degree must be the lowest in the whole graph. Since we want to get the smallest possible graph, $deg(v) = 2$. Therefore, the degrees of the nodes in the island components G_1 and G_2 are at least 3. However, only one node per island component can be connected to c, therefore, G_1 and G_2 contain at least 4 nodes. □

Lemma 12. *Let G be an α-critical graph containing an α-critical component CC. Let also $c_1 \in CC$ and $c_2 \in CC$ be the α-critical nodes that G_1 and G_2 respectively are connected to, $G_1^* = G_1 + c_1$ and $G_2^* = G_2 + c_2$. Finally, let G_1' and G_2' denote the graphs resulting by adding* *one connected α-critical node (from CC) at a time to G_1^* and G_2^* until the resulting graphs cannot be extended any further without violating the α-density constraint. G_1' overlaps with G_2'.*

Proof. By Lemma 10, we know G_1^* and G_2^* are α-dense.

Assume G_1' and G_2' do not overlap, i.e. we have

$$|E_1'| + 1 < \frac{\alpha|V_1'|(|V_1'|+1)}{2} \tag{3.16}$$

$$|E_2'| + 1 < \frac{\alpha|V_2'|(|V_2'|+1)}{2} \tag{3.17}$$

Since G_1' and G_2' do not overlap, we have $T \geq 1$ α-critical nodes not absorbed by either of G_1' and G_2'. Moreover, $|E| = |E_1'| + |E_2'| + T - 1$, $|V| = (|V_1'| + |V_2'| + T)$. Finally, for the sake of simplicity, let $K = |V_1'| + |V_2'|$. The density of G is calculated as follows:

$$\begin{aligned} d(G) &= \frac{2|E|}{|V||V-1|} \\ &= \frac{2(|E_1'| + |E_2'| + T - 1)}{(K+T)(K+T-1)} \\ &< \frac{2(\frac{\alpha|V_1'|(|V_1'|+1)}{2} + \frac{\alpha|V_2'|(|V_2'|+1)}{2} + T - 1)}{(|V_1'|+|V_2'|+T)(|V_1'|+|V_2'|+T-1)} \\ &= \alpha\left[\frac{|V_1'|(|V_1'|+1) + |V_2'|(|V_2'|+1) + \frac{2(T-1)}{\alpha}}{|V_1'|(K+T-1) + |V_2'|(K+T-1) + T(K+T-1)}\right] \end{aligned} \tag{3.18}$$

We have, by the assumption of no overlap, $T \geq 1$. Moreover, we know from Lemma 11 that $|V_1'| \geq 4$ and $|V_2'| \geq 4$, which means $K = |V_1| + |V_2| \geq 8$. Putting it all together, we have

$$\begin{aligned} |V_1'|(|V_1'|+1) &< |V_1'|(K+T-1) \\ |V_2'|(|V_2'|+1) &< |V_2'|(K+T-1) \end{aligned} \tag{3.19}$$

Moreover,

$$\frac{2(T-1)}{\alpha} < T(K+T-1)$$

Since $\alpha \geq \frac{1}{3}$, we have

$$\begin{aligned} \frac{2}{3}(T-1) &< T(8+T-1) \\ \Leftrightarrow \frac{2}{3} &< T\left(1+\frac{8}{T-1}\right) \end{aligned} \tag{3.20}$$

Since $T \geq 1$, 3.20 is always satisfied.

Overall, by 3.19 and 3.20, for every term in the nominator of the ratio within the square brackets of 3.18, we have a larger term in the denominator. Hence the nominator is smaller than the denominator, i.e. we have $d(G) < \alpha$, which contradicts with the fact that G is α-dense. Therefore, G'_1 and G'_2 overlap. □

Corollary 2. *G'_1 and G'_2 are α-strongly connected.*

Proof. By Lemma 9, G_1 is at least $\frac{1}{2}$-dense and therefore by Lemma 7 G_1 is α-strongly connected. By definition, G'_1 is constructed from G_1 adding a one node at a time such that no more extension is possible. Hence, by construction, G'_1 is α-strongly connected. G'_2 is covered by symmetry. □

Although, we now have the sufficient material for designing an algorithm for mining DCBs for $\frac{1}{3} \leq \alpha \leq 1$, we make the following final observation, which we will use for developing strategies for a more efficient algorithm.

Lemma 13. *Let $\frac{1}{3} \leq \alpha < \frac{1}{2}$ and $G = (V, E)$ be a connected α-critical graph containing leaf components $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. If $|V_1| = |V_2|$, then G_1 and G_2 are $(2\alpha - \frac{1}{3(|V_1|-1)})$ quasi-cliques. If $|V_1| < |V_2|$, G_1 is a (2α)-quasi clique and G_2 is a $(\frac{3\alpha}{2} - \frac{1}{3(|V_2|-1)})$-quasi-clique, respectively.*

Proof. By Lemma 3 any node $v \in V_i$, $i \in 1, 2$, is connected to more than $\alpha(|V|-1)$ nodes. Since, v can be connected to at most one α-critical node, v is connected more than $\alpha(|V|-1) - 1$ nodes within G_i. We have two cases: $|V_1| = |V_2|$ and $|V_1| < |V_2|$ (the case $|V_1| > |V_2|$ is covered by symmetry):

- *Case* $|V_1| = |V_2|$*:*
  The degree of each node $v \in V_1$ is at least

$$\begin{aligned} deg_{G_1}(v) &\geq \alpha(|V|-1)-1 \\ &= \alpha(|V_1|+|V_2|+|CN(G)|-1)-1 \\ &= (2\alpha - \frac{1}{3(|V_1|-1)})(|V_1|-1)+\alpha(|CN(G)|+1)-\frac{2}{3} \end{aligned}$$

  Since $\alpha \geq \frac{1}{3}$ and $|CN(G)| \geq 1$, we have $\alpha(|CN(G)|+1)-\frac{2}{3} \geq 0$. Therefore, $deg_{G_1}(v) \geq (2\alpha - \frac{1}{3(|V_1|-1)})(|V_1|-1)$, which means G_1 and G_2 (by symmetry) are at least $(2\alpha - \frac{1}{3(|V_1|-1)})$-quasi-cliques.

- *Case* $|V_1| < |V_2|$.
  Then, degree of each node in G_1 (or G_2) is at least $\alpha(|V|-1)-1$. We analyze G_1 and G_2 separately.

  - *Properties of* G_1*:*
    Since $|V_1| < |V_2|$, the degree of $v \in V_1$ is

$$\begin{aligned} deg_{G_1}(v) &\geq \alpha(|V|-1)-1 \\ &= \alpha(|V_1|-1)+\alpha(|V_2|-1)+\alpha(|CN(G)|+1)-1 \\ &\geq 2\alpha(|V_1|-1)+\alpha(|CN(G)|+2)-1 \end{aligned} \tag{3.21}$$

    Since $\alpha \geq \frac{1}{3}$ and $|CN(G)| \geq 1$, we get $\alpha(|CN(G)|+2)-1 \geq 0$. Therefore $deg_{G_1}(v) \geq 2\alpha(|V_1|-1)$. This means the smaller leaf component G_1 is a (2α)-quasi-clique.

  - *Properties of* G_2*:*
    By Lemma 8, we have $|V_2| < 2|V_1|$ or $|V_2| \leq 2|V_1|-1$, which implies $(|V_1|-1) \geq \frac{(|V_2|-1)}{2}$. Therefore, the degree of $v \in G_2$ is

$$\begin{aligned} deg_{G_2}(v) &\geq \alpha(|V|-1)-1 \\ &= \alpha(|V_1|-1)+\alpha(|V_2|-1)+\alpha(|CN(G)|+1)-1 \\ &\geq \frac{3\alpha}{2}(|V_2|-1)+\alpha(|CN(G)|+1)-1 \\ &= (\frac{3\alpha}{2}-\frac{1}{3(|V_2|-1)})(|V_2|-1)+\alpha(|CN(G)|+1)-\frac{2}{3} \end{aligned} \tag{3.22}$$

Since $\alpha \geq \frac{1}{3}$ and $|CN(G)| \geq 1$, we get $\alpha(|CN(G)|+1)-\frac{2}{3} \geq 0$. Hence, $deg_{G_2}(v) \geq (\frac{3\alpha}{2} - \frac{1}{3(|V_2|-1)})(|V_2|-1)$. This means the larger leaf component G_2 is a $(\frac{3\alpha}{2} - \frac{1}{3(|V_2|-1)})$-quasi-clique.

$\square$

In summary, we showed that the DCB problem is a computational hard problem. However, we derived several important properties that a DCB must have in order to satisfy the DCB constraint. Our most important findings are: (1) the DCB constraint is loose anti-monotone for $\frac{1}{2} \leq \alpha \leq 1$ and (2) a DCB must have restricted topology if it is not α-strongly-connected, $\frac{1}{3} \leq \alpha < \frac{1}{2}$. As we shall we in the next chapter, these findings are the key ingredients in the design of the DCB-Miner algorithm.

Chapter 4

Algorithm

In this section we describe a novel pattern mining approach based algorithm to solve the DCB problem and prove its completeness. As surveyed in [24], typical data mining problems such as finding clusters, patterns and correlations, are undecidable problems. Therefore, they are computationally hard problems. Nowadays typical amount of data to be mined, such as transactional databases and high throughput experimental data, is increasing with a speed higher than ever. This makes the problem even harder. However, the theoretical worst case almost never happens in real datasets and algorithms that are based on *search space prunning strategies* succeed to efficiently solve data mining problems. For example, the theoretic worst case, i.e. exponential runtime, for the famous *Apriori algorithm* in [3] can be easily constructed. However, experience shows that such data never occurs in real life. Similarly, we showed that DCB problem is $\mathcal{NP}$-hard. However, we demonstrate in the following that, on the biological instances at hand, the problem becomes tractable by carefully designing a search space pruning strategy. Although, exponential in the worst case, as we show in the experiments section, it is quite efficient for solving the DCB problem, especially when given parameter settings that are close to those of the real functional modules.

We start with detailing the algorithm to show that it is *correct*, i.e. it finds only the maximal DCBs and then proceed to prove that it is *complete*, i.e. it finds all maximal DCBs satisfying the constraints specified by the input parameters. Finally, we will describe a heuristic post-processing step that summarizes the result set of DCBs.

4.1 Density Constrained Bicluster Miner (DCB-Miner) Algorithm

We propose a three phase algorithm called Density Constrained Bicluster Miner (DCB-Miner). The core strategy of the algorithm is to drastically narrow down the exponential search space by means of the loose anti-monotonicity and other properties of DCB constraint that are derived in Section 3. The basic idea is that we imagine that all subnetworks are organized into a hierarchical structure (formally a lattice) where a subgraph is a child of another one if it can be obtained by adding exactly one neighbor node and the corresponding edges to the parent subgraph. Note that a child is larger than its parent which may be a bit counterintuitive. Accordingly, the complete interaction network at hand can be identified with the bottom of this structure. At the top of this structure are all pairs of genes in the interaction network (See Figure 4.1 for an example). Then we mine this structure in three phases as follows:

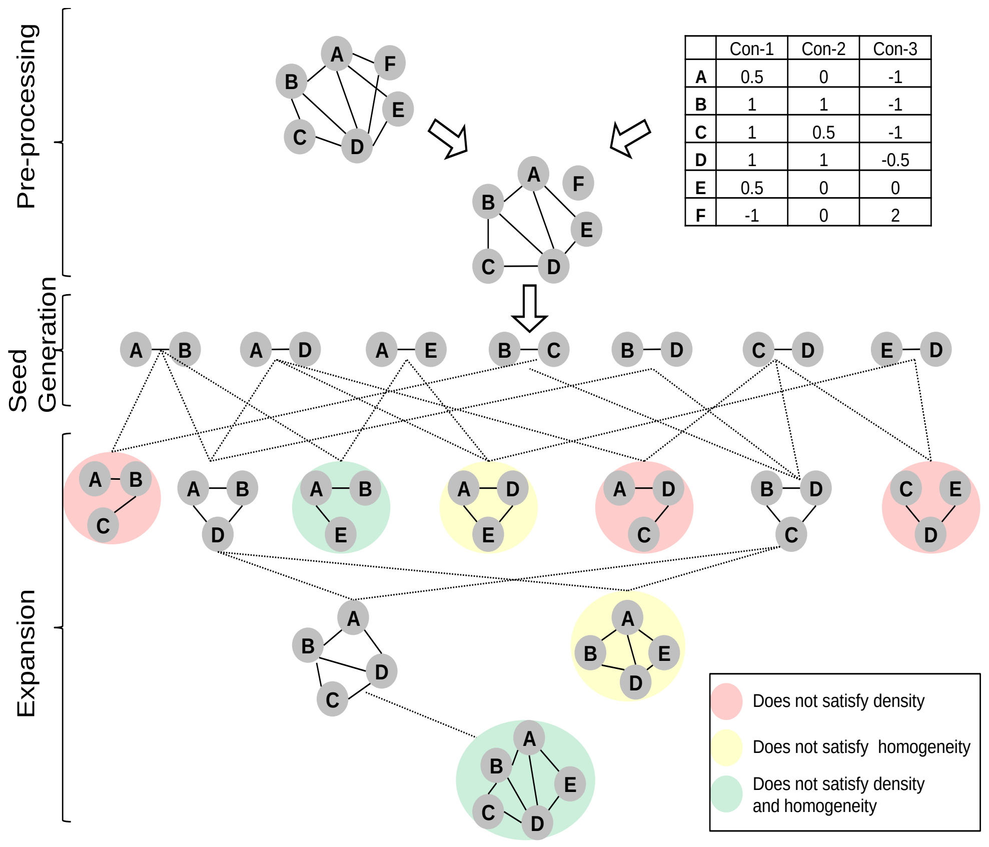
- *Expand-by-one*: First, we traverse the explained structure top-down, in a *breadth-first* (level-wise) manner. This means that subnetworks of size n are only checked upon having checked all subnetworks of size $n-1$. The trick is that when it comes to examining subnetworks of size n, we can restrict ourselves to checking children of DCBs of size $n-1$, as the loose anti-monotonicity of the DCB constraint guarantees that every DCB of size n necessarily has a DCB of size $n-1$ as a parent. In the first iteration, this results in removing all pairs of genes that are either not connected or not sufficiently co-expressed (violating the homogeneity constraint). Then, in the second iteration, we check only children of connected, co-expressed pairs of genes and, again, keep only 3-gene-DCBs to go to the level of 4-gene DCBs and so on. If the underlying network is sufficiently sparse, which applies for the graphs at hand, this greatly speeds up the search.

- *Merge*: Second, in the case of $\frac{1}{3} \leq \alpha < \frac{1}{2}$, *Expand-by-one* phase may fail to find a DCB if it is not *alpha*-strongly-connected. However, such a DCB has a special topology, i.e. its overlapping subgraphs must be found in the *Expand-by-one* phase. Hence, in the *Merge* phase, we check for such graphs by making use of the overlapping property along with many other constraints derived in Section 3.3.2.

- *CheckMaximality* : Finally, the patterns found from *Expand-by-one* and *Merge* phases are checked for maximality. This phase is necessary, because the DCB constraint is loose anti-monotone for $\alpha \geq \frac{1}{2}$. This means that, some of the permutations of the nodes of a DCB may fail to be α-strongly-connected. Those permutations cause *Expand-by-one* phase to identify them as DCBs although they are not maximal (See Figure 4.1). Moreover, in the *Merge* phase a newly found pattern may can contain one or both of the overlapping DCBs, which are clearly not maximal.

The pseudo code of the DCB-Miner algorithm is given in Algorithm 4.1. The algorithm starts with a *preprocessing phase*, in which non-homogenous edges, i.e. edges between nodes that do not satisfy homogeneity constraint, are removed from the input graph. This is because homogeneity constraint is anti-monotone and therefore any super-graph containing of the 2-node non-homogenous DCB will be non-homogenous. This usually results in a partitioning of the attributed graph. In the reduced graph we identify all connected components and mine them independently. Note that due to the independence of connected components, mining of the components can be done in a *parallel* way. This can be useful in case of large input graphs and availability of parallel computation resources. After preprocessing, DCB-Miner generates a DCB for every connected pair of nodes, which are also called the *seeds*. These 2-node DCBs are then fed into the *Expand-by-one* subroutine.

The pseudo code of the subroutine *Expand-by-one* is given in Algorithm 4.2. It takes as input a set of DCBs and returns maximally expanded-by-one DCBs. Let *level* denote the number of nodes of the current DCBs, which is 2 at the start. At each level, we expand existing DCBs of size *level* by all neighboring nodes, one at a time. Note that for every expanded pattern, the corresponding homogenous feature subspace is uniquely determined. This procedure helps us to minimize the generation and testing of candidates as much as possible. Although the details are not shown, we do an incremental testing of homogeneity. This means that if $curPattern = (V, E, D, \mathcal{A})$ and $candidatePattern = curPattern + v$, then we only check whether or not $candidatePattern$ is homogenous on dimensions $d \in D$. This is because, as noted in the proof of anti-monotonicity of homogeneity constraint (See Theorem-2), extension of a valid DCB can only decrease the cardinality of the homogenous subspace. Moreover, if an expanded graph fulfills the DCB constraints, then we know that the parent pattern is not maximal and we discard the parent pattern. Otherwise, i.e. none of the neighbors of a the $curPattern$ results in a larger DCB, then we add $curPattern$ to the result set as it cannot be extended further in the following iterations. The result set


Pre-processing
Seed Generation
Expansion
Does not satisfy density
Does not satisfy homogeneity
Does not satisfy density and homogeneity

	Con-1	Con-2	Con-3
A	0.5	0	-1
B	1	1	-1
C	1	0.5	-1
D	1	1	-0.5
E	0.5	0	0
F	-1	0	2

Figure 4.1: Illustration of the DCB-Miner on a toy example consisting of six genes with logged-2 fold changes across three experimental conditions. We assume the parameters are: $\alpha = 0.8$, $\theta_h = 0.5$ and $\theta_{dim} = 2$, i.e. we want to find modules having density 0.8, whose corresponding genes are co-expressed (i.e. within a fold change range of $2^{0.5} \approx 1.4$) across at least 2 experimental conditions. The algorithm starts with (1) the *Preprocessing* step in which the edges A-F and D-F are removed because genes pairs A-F and D-F satisfy the fold change range constraint across only 1 and 0 conditions respectively. (2) Then, we generate 2-node *seed* modules for every remaining edge. (3) Next, the iterative *Expand-By-One* step is called. In every iteration, current modules are expanded by a neighboring gene. For example, the expansion of the module A-D results in the larger candidate modules A-B-D, A-C-D and A-D-E. Note that, the candidate A-C-D does not satisfy the density constraint, because it has density $\frac{2}{3} < 0.8$, so it is pruned. Moreover, A-D-E is also pruned because it has only one homogeneous condition and does not satisfy the co-expression homogeneity constraint. In this iteration only subnetworks A-B-D and B-C-D satisfy all DCB constraints and the rest (and their super-networks) are pruned from the search space. During the next iteration candidate modules A-B-C-D and A-B-D-E are generated. Only A-B-C-D satisfy the DCB constraints. A-B-C-D is returned as a maximal DCB with the associated context consisting of conditions 1 and 3. To summarize, the constraint based prunning strategy reduces the search space from $2^6 = 64$ to 17.

Algorithm 4.1 DCB-Miner: Densely Constrained Bicluster Miner

```
1: INPUT: Attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$,
       Density threshold $\alpha$,
       Homogeneity threshold $\theta_h$,
       Minimum dimensionality $\theta_{dim}$
2: OUTPUT: The complete set of maximal DCBs
3: \\ PREPROCESSING: remove non-homogenous edges from $\mathcal{G}$
4: for all (edges $e = \{v_1, v_2\} \in \mathcal{E}$) do
5:    $G_e \leftarrow (\{v_1, v_2\}, \{\{v_1, v_2\}\})$
6:    if $G_e$ does not satisfy homogeneity constraint then
7:       $E \leftarrow E - \{\{v_1, v_2\}\}$
8: \\ Run in parallel
9: for all (connected components $G_i = (V_i, E_i)$ in $\mathcal{G}$) do
10:    $currPatterns_i \leftarrow \emptyset$
11:    $criticalPatterns_i \leftarrow \emptyset$
12:    \\ Generate 2-node DCBs
13:    for all (edges $e = \{v_1, v_2\} \in E_i$) do
14:       $G_2 \leftarrow (\{v_1, v_2\}, \{\{v_1, v_2\}\}, D)$
15:       $currPatterns_i.add(G_2)$
16:    \\FIRST PHASE: expand-by-one
17:    $currPatterns_i \leftarrow Expand\text{-}by\text{-}one(currPatterns_i)$
18:    if $(\alpha < \frac{1}{2})$ then
19:       \\ SECOND PHASE: merge
20:       $criticalPatterns_i \leftarrow Merge(currPatterns_i)$
21:       $currPatterns_i.addAll(Expand\text{-}by\text{-}one(criticalPatterns_i))$
22:    \\ THIRD PHASE: remove non-maximal patterns from $currPatterns$
23:    $resultSet.addAll(CheckMaximality(currPatterns_i))$
24: return $resultSet$
```

is a organized as a *stack* data structure within the *resultSet*, such that when the *Expand-by-one* finishes, larger patterns are on the top of stack whereas the smaller ones will be at the bottom of the stack. This will prove to be a useful strategy in maximality check step, which we discuss later in this section. After having considered all patterns of a certain level (size), we move to the next iteration and continue until all patterns are maximally expanded-by-one. Therefore, by design the *Expand-by-one* does a breadth-first search. The advantage of this search is that at any point of time we only need to keep valid patterns of two levels in memory. This reduces the amount of memory needed substantially and also eliminates multiple generation and testing of candidate DCBs.

At this point, it is necessary to discuss a couple of implementation details regarding the *Expand-by-one* phase. First, the *validCandidates* data structure is a hash table. It uses the ID of a valid pattern as the key for hashing, where ID of a pattern is the string formed

Algorithm 4.2 First phase: Expand-by-one

```
1: INPUT: $currDCBs$, the set of current DCBs
2: OUTPUT: the set $resultSet$ of maximally expanded-by-one DCBs
3: $resultSet \leftarrow \emptyset$
4: \\ Proceed to the next level
5: while ($curPattern() \neq \emptyset$) do
6:    $validCandidates \leftarrow \emptyset$
7:    while ($curPattern \leftarrow currDCBs.pop() \neq NULL$) do
8:       $curPattern.isExtensible \leftarrow false$
9:       for all (neighboring nodes $v$ of $curPattern$) do
10:         $candidatePattern \leftarrow curPattern + v$
11:         if ($validCandidates$ does not contain $candidatePattern$ AND
            $candidatePattern$ satisfies DCB constraints) then
12:            $validCandidates.push(G + v)$
13:            $G.isExtensible \leftarrow true$
14:      if ($G.isExtensible = false$) then
15:         $resultSet.add(G)$
16:   $currDCBs \leftarrow validCandidates$
17: return $resultSet$
```

by concatenation of sorted IDs of its member nodes separated by the '-' character. Hence, IDs of two graphs are different unless they contain the same set of nodes. Secondly, in its current form *Expand-by-one* may test an invalid candidate pattern, i.e. not satisfying the DCB constraint, multiple times. This is not the case for valid candidate patterns because once a candidate is found to be a valid pattern, it is saved in $validCandidates$. In case of regeneration of the same valid pattern, the algorithm first checks the $validCandidates$ hash map and therefore the candidate will not be checked against the DCB constraint again. In principle, we could have a similar data structure for invalid candidates so as to eliminate multiple check of on invalid candidates. However, as expected, we observed that the number of invalid candidates is typically much more than valid candidates and such a data structure occupies large amount of memory, which slows down the algorithm in practice. We found that our strategy is more efficient than keeping track of invalid candidates.

Recall that in Theorem 5, we showed that for $\alpha \geq \frac{1}{2}$, a connected α-dense graph is α-strongly-connected. This requires existence of at least one permutation of its nodes such that by following this permutation the graph can be generated by adding one connected node at a time to the current graph starting from the first node in the permutation. Indeed that is what *Expand-by-one* does, i.e. tries all valid permutations and stops expanding a permutation when DCB constraint is not met. Hence, by design, it can find all α-strongly-connected graphs. However, if $\frac{1}{3} \leq \alpha < \frac{1}{2}$, then a connected α-dense graph G may not be α-strongly-connected, i.e. G is α-critical. In this case, the graph will be missed by the *Expand-by-one* phase. However, in Section 3.3.2 we showed that, if this is the case, then G must have a certain topology, i.e. it contains two overlapping α-strongly-connected subgraphs G_1^{EBO} and G_2^{EBO}, where EBO means maximally expanded-by-one (See Figure 4.3 for an example). This means that even if the *Expand-by-one* phase misses the pattern G, it is able to find G_1^{EBO} and G_2^{EBO}. The *Merge* phase make use of this observation along with other ones derived Section 3.3.2 in order to find G.

The pseudo code of *Merge* phase is given in Algorithm 4.3. As input we have a set of maximally expanded-by-one DCBs. First, we determine the pairs of DCBs that overlap by at least one node. They must differ in at least two nodes, otherwise one would be found from the other in the *Expand-by-one* phase. Assume the pair G_1^{EBO} and G_2^{EBO} overlaps and also satisfies the aforementioned two conditions. We determine all α-quasi-cliques G_1 contained in G_1^{EBO} and G_2 in G_2^{EBO} and apply the online α-quasi-cliquishness bounds derived in Lemma 13. For all pairs G_1 and G_2 satisfying the conditions, we determine all simple paths between them such that the nodes of the path have to be in $\{V_1^{EBO} \cup V_2^{EBO}\} \setminus \{V_1 \cup V_2\}$. There is one crucial design decision that needs to be addressed. Although, not shown explicitly for the sake of simplicity, in its present form *Merge* phase calls a modified version of *Expand-by-one* subroutine in order to find α-quasi-cliques, i.e. G_1 and G_2 to be merged. The modification is that *Expand-by-one* only mines G_1^{EBO} and G_2^{EBO} but not the complete graph. This is indeed duplication of work, i.e. G_1 and G_2 were already be found in early iterations of *Expand-by-one*. However, recall that, *Expand-by-one* only keeps maximally expanded-by-one DCBs and hence instead of G_1 and G_2 we have G_1^{EBO} and G_2^{EBO} in the a result of *Expand-by-one*. Alternatively, we can store G_1 and G_2 along the way in a separate data structure. However, the case of α-critical graph hardly happens. Hence, we have a trade of between memory and run time. We can save from run time by storing the G_1 and G_2 to be used in *Merge* phase directly at the cost of increased memory consumption. Alternatively, we can mine them on the fly from G_1^{EBO} and G_2^{EBO}. We tried both alternatives and did not observe a significant difference.

Note that we call the *Expand-by-one* subroutine once more on the graphs found in the merge phase. Although we could not find an example of such a graph, theoretically it can exist. However, if it exists, it must be found by the second call of the *Expand-by-one* phase.

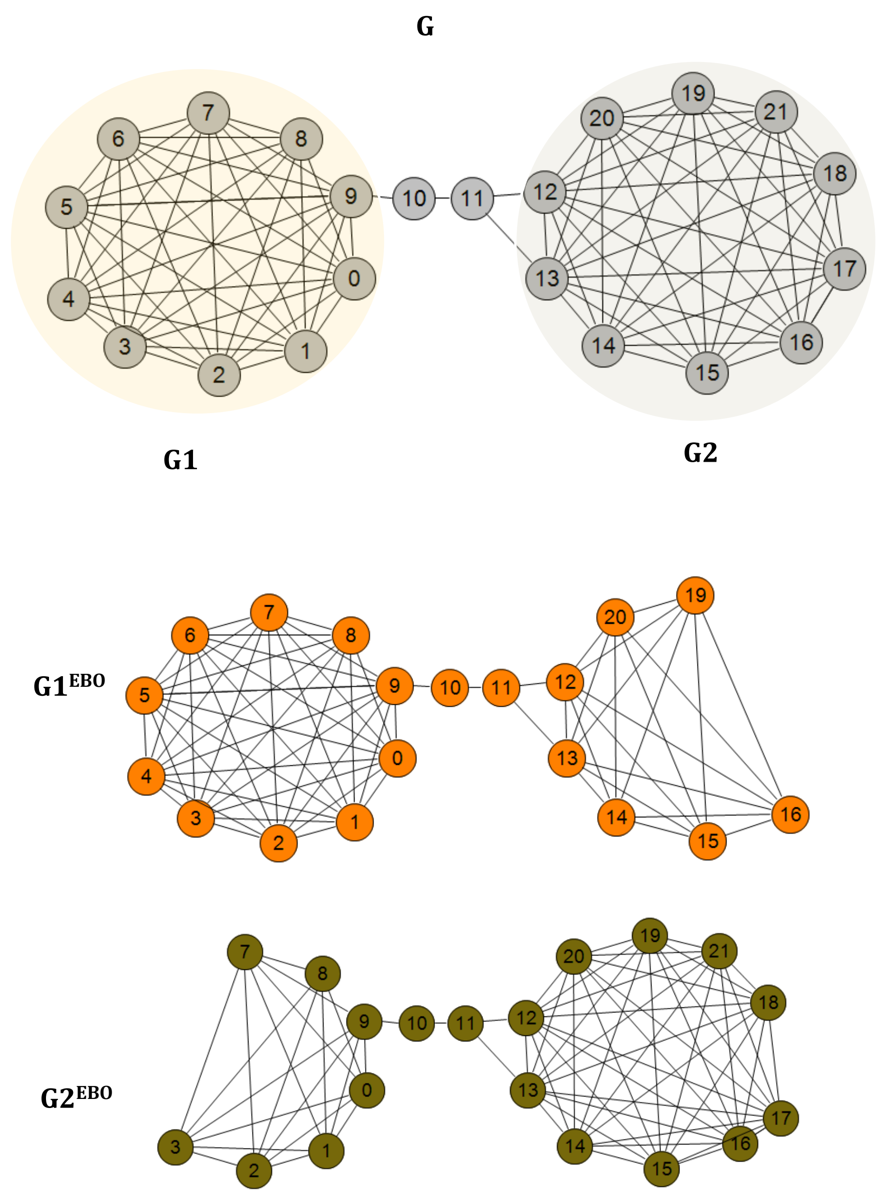


Figure 4.2: Graph G contains two α-critical nodes (nodes 10 and 11) for $\alpha = 0.406$. Hence, G cannot be found in the *Expand-by-one* phase . However, G contains at least two α-strongly-connected overlapping subgraphs, i.e. G_1^{EBO} and G_2^{EBO}, which are guaranteed to be found in the *Expand-by-one* phase. The *Merge* phase mines α-quasi-cliques of G_1^{EBO} and G_2^{EBO}. In this particular example, G_1 and all its subgraphs are at least 0.406-quasi-clique of G_1^{EBO}. Similarly, G_2 and all its subgraphs are at least 0.406-quasi-clique of G_2^{EBO}. Hence, in the *Merge* phase simple paths will be searched between all 0.406-quasi-cliques of G_1^{EBO} and G_2^{EBO}, which in this case consists of nodes 10 and 11.

The reason will be clear when we discuss the proof of the *completeness* of DCB-Miner.

Algorithm 4.3 Second Phase: Merge

```
1: INPUT: $currDCBs$, the set of maximally expanded-by-one DCBs
2: OUTPUT: $resultSet$, the set of $\alpha$-critical graphs
3: for all (overlapping pairs of graphs $G_1^{EBO}, G_2^{EBO} \in currDCBs$
   such that $|V_1^{EBO} - V_2^{EBO}| \geq 2$ AND $|V_2^{EBO} - V_1^{EBO}| \geq 2$) do
4:   for all (non-overlapping $\alpha$-quasi-cliques $G_1 \subset G_1^{EBO}$, $G_2 \subset G_2^{EBO}$) do
5:     \\ Prunning based on $\alpha$-quasi-cliquishness
6:     if ($|V_1| = |V_2|$) then
7:       if ($G_1$ and $G_2$ are not at least $2\alpha$-quasi-cliques then
8:         continue;
9:     else
10:      if ($|V_1| < |V_2|$) then
11:        if ($G_1$ is not at least $(2\alpha)$-quasi-clique OR
           $G_2$ is not at least $(\frac{3\alpha}{2} - \frac{1}{3(|V_2|-1)})$-quasi-clique) then
12:          continue;
13:      else
14:        if ($G_2$ is not at least $(2\alpha)$-quasi-clique OR
           $G_1$ is not at least $(\frac{3\alpha}{2} - \frac{1}{3(|V_1|-1)})$-quasi-clique) then
15:          continue;
16:    \\ Search for paths
17:    for all (simple paths $P \subset (\{V_1^{EBO} \cup V_2^{EBO}\} \setminus \{V_1 \cup V_2\})$ connecting $G_1$ and $G_2$)
       do
18:      if ($G \leftarrow G_1 \cup P \cup G_2$ fulfills DCB constraints and $G$ is $\alpha$-critical) then
19:        $resultSet.add(G)$
20: return $resultSet$
```

The final step in the DCB-Miner algorithm is the maximality check (Algorithm 4.4), which removes non-maximal DCBs from the result set. This postprocessing step is necessary, because the DCB constraint is loose anti-monotone for $\alpha \geq \frac{1}{2}$. This means that, at least one permutation τ_1 exists such that the permutation induces a chain of subgraphs satisfying the DCB constraint. *Expand-by-one* is able to construct the complete DCB by adhering to τ_1 . For other permutations, say τ_2, which are also analyzed by *Expand-by-one* phase, the permutation may fail to generate the full graph and therefore may result in a subgraph rather than the complete graph. (See Figure 4.3 for an example) Therefore, such subgraphs must be eliminated from the final result. The maximality check step makes use of a statistic array *occurence*, which keeps the number of patterns each node is included in. Moreover, we have the $maximalPatterns$. The key of this hash table is a node id, and the corresponding

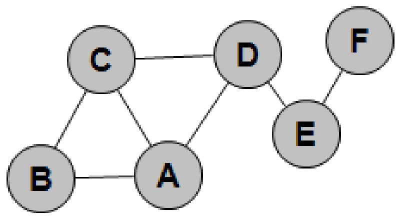


Figure 4.3: A sample graph that illustrates the necessity of the maximality check step. The graph is $\frac{7}{15}$-dense. For $\alpha = \frac{7}{15}$, the graph is α-strongly-connected, i.e. the permutation $\{A, B, C, D, E, F\}$ induces a chain of connected α-dense graphs. However, the permutation $\{F, E, D, A, B, C\}$ fails to induce such a chain after extending node permutation $\{F, E, D, A\}$ with node B. Therefore, the graph $\{F, E, D, A\}$ is a maximally expanded-by-one graph even though it is not a maximal α-dense graph.

bucket stores all DCBs that contain this particular node. As highlighted in the explanation of *Expand-by-one* phase, the *currDCBs* is a stack data structure such that larger DCBs are on top and smaller DCBs are at the bottom. Hence, when inserting new patterns, if the DCB is not maximal, the maximal DCB will already be in the *maximalPatterns*. Hence, we just move the next DCB. After inserting finishes, *maximalPatterns* contains only maximal DCBs. However, a DCB is stored in each bucket corresponding to each of its nodes. Hence, we do one more pass over *maximalPatterns* by inserting its elements into the set *resultSet*, which eliminates the duplicates.

Note that DCB-Miner is *correct* by design, i.e. it returns only maximal DCBs. This is because at each iteration of *Expand-by-one* phase, candidates are retained only if they satisfy the DCB constraint (See line 11 of Algorithm 4.2). Similarly, in the *Merge* phase, newly found patterns are checked against the DCB constraint (See line 18 of Algorithm 4.3). Finally, the *CheckMaximality* phase assures that non-maximal DCBs are filtered out. Hence, DCB-Miner is *correct.* Next, we analyze its completeness.

4.2 Completeness

Theorem 7. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ be an attributed graph. If $G = (V, E, D, \mathcal{A})$, $V \subseteq \mathcal{V}, E \subseteq \mathcal{E}, D \subseteq \mathcal{D}$ is a DCB, then G is contained in the result set of DCB-Miner.*

Algorithm 4.4 Third Phase: CheckMaximality

```
1: INPUT: $currDCBs$, the set of current DCBs
2: OUTPUT: $resultSet$, the set of maximal DCBs
3: \\ Initialize occurrence statistics
4: for $i \leftarrow 0$ to $|\mathcal{V}|$ do
5:    $occurence \leftarrow$ number of DCBs including node $v_i$
6: $maximalPatterns \leftarrow \emptyset$
7: \\ Insert DCBs into buckets
8: for all ($G = (V, E, D, \mathcal{A})$ of $currDCBs$) do
9:    $v \leftarrow \arg\min_{v \in V} occurence(v.getID())$
10:   $bucket_v \leftarrow maximalPatterns.getBucketOf(v)$
11:   $isMaximal \leftarrow true$
12:   for all ($insertedDCB$ in $bucket_v$) do
13:      if ($insertedDCB$ is a super-pattern of $G$) then
14:         $isMaximal \leftarrow false$
15:   if ($isMaximal$) then
16:      for all (nodes $v'$ in $nextDCB$) do
17:         $bucket_{v'} \leftarrow maximalPatterns.getBucketOf(v')$
18:         $bucket_{v'}.insert(G)$
19: \\ All DCBs are maximal
20: for all ($G = (V, E, D, \mathcal{A})$ of $maximalPatterns$) do
21:   $resultSet.insert(G)$;
22: return $resultSet$
```

Proof. We distinguish between two ranges for α.

- $\frac{1}{2} \leq \alpha \leq 1$. Since $\alpha \geq \frac{1}{2}$, by Lemma 7, G is α-strongly connected. By definition, α-strong connectedness requires existence of at least one permutation of type described in Definition 9. Since *Expand-by-one* , starting from 2-nodes pairs, checks all permutations, it is guaranteed to find all DCBs.

- $\frac{1}{3} \leq \alpha < \frac{1}{2}$. If G is α-strongly connected, we use the same argument as in the first case. Otherwise, let $n = |V|$ and $G_n = G$ be a DCB which is not α-strongly connected. We can decompose G by removing one node at a time as follows. Let v_n be an α-removable node in G_n which is not a bridge node, i.e. $G - v_n = G_{n-1}$ is a DCB. We apply this strategy recursively until G_j, $j \leq n$, contains an α-critical node, i.e. no such v_j exists. By Theorem 6, G_j contains two island leaf components G_1 and G_2 and one α-critical component CC, i.e. $G_j = G_1 \cup CC \cup G_2$. G_1 and G_2 are at least $\frac{1}{2}$-dense by Lemma 10 and therefore by Lemma 7 they are α-strongly connected.

All α-strongly connected DCBs are discovered in the first call of the *Expand-by-one* phase. However, the algorithm keeps only the maximal ones, i.e. *Expand-by-one* may output supergraphs of G_1 and G_2, say G_1^{EBO} and G_2^{EBO} respectively. As discussed in the *Merge* phase, G_1 and G_2 are guaranteed to be found from G_1^{EBO} and G_2^{EBO} since they are both α-strongly connected. Once we identify G_1 and G_2, G_j is found by checking the simple paths in $\{V_1^{EBO} \cup V_2^{EBO}\} \setminus \{V_1 \cup V_2\}$. Once, we have G_j, the second call to *Expand-by-one* is now able to find the complete graph G. This is because, by construction G_j is obtained from G by removing one node at a time such that the resulting graph is connected and α-dense. Once, we have G_j, *Expand-by-one* does the exact opposite operation of decomposition and therefore is able to find G.

□

4.3 Post-processing

A major challenge in subspace clustering and pattern mining algorithms is the size of the result set. Recently is has been shown that compression of the frequent itemsets, i.e. selection of an optimal set of representative patterns, is $\mathcal{NP}$-hard and a heuristic pattern compression approach is proposed for this problem [83]. Unfortunately, DCB-Miner is not immune to this problem either. This is justified by the fact that currently available PPI/GI networks are far from being complete, meaning that a substantial amount of genes, proteins and interactions are still missing. Moreover, the gene expression experiments contain a high amount of noise. These issues result in a significant amount of modules that are split up into fractions.

The post-processing step, which is described in Algorithm 4.5, is a heuristic that solves this problem to a significant degree by relaxing the density and homogeneity constraints in such cases. In a refinement procedure, we iteratively merge pairs of DCBs if they overlap in at least 75% of their members as well as in at least 80% of their associated subspaces. These values were found empirically. Note that, simply relaxing the parameter of DCB-Miner would not solve the problem, as it would not solve the problem in the context of frequent itemset mining. This is because starting with relaxed parameters will give much more patterns at the end. On the contrary, we need compression of patterns and we achieve this by first finding high quality patterns and then iteratively merging them. As we will see in Chapter 5, the post processing step succeeds to compress pattern significantly at a cost of increased runtime and possible decrease in the quality of biclusters as a result of relaxed constraints, i.e. lower density. Note that the output of the post-processing phase is dependent on the order that the modules are processed. However, we observed that this does not result in any significant difference in terms of the output.

Algorithm 4.5 Post-processing

```
1: INPUT: $currDCBs$, the set of current DCBs
       Member merge threshold $mmt$,
       Subspace merge threshold $smt$
2: OUTPUT: $masterModules$, the set of merged DCBs
3: $mergeOccured \leftarrow true$
4: while $mergeOccured$ do
5:   $mergeOccured \leftarrow false$
6:   $masterModules \leftarrow \emptyset$
7:   for $i = 1$ to $currDCBs.size()$ do
8:     $bestMatch \leftarrow \emptyset$
9:     $bestOverlapScore \leftarrow 0$
10:    $firstModule \leftarrow currDCBs.get(i)$
11:    for $j = i + 1$ to $currDCBs.size()$ do
12:      $j \leftarrow j + 1$
13:      $secondModule \leftarrow currDCBs.get(j)$
14:      $memberOverlap \leftarrow firstModule.getMemberOverlap(secondModule)$
15:      $subspaceOverlap \leftarrow firstModule.getSubspaceOverlap(secondModule)$
16:      if $memberOverlap \geq mmt$ and $subspaceOverlap \geq smt$ AND
         $memberOverlap * subspaceOverlap > bestOverlapScore$ then
17:        $bestOverlapScore \leftarrow memberOverlap * subspaceOverlap$
18:        $bestMatch \leftarrow secondModule$
19:    if $bestMatch \neq \emptyset$ then
20:      $masterModules.insert(firstModule.merge(bestMatch))$
21:      $firstModule.setMaximal(false)$
22:      $bestMatch.setMaximal(false)$
23:      $mergeOccured \leftarrow true$
24:    $i \leftarrow i + 1$
25:  \\ Remove non-maximal modules
26:  for all modules $module$ in $currDCBs$ do
27:    if $module.isMaximal()$ then
28:      $masterModules.insert(module)$
29:  $currDCBs \leftarrow masterModules$
30: return $masterModules$
```

Chapter 5

Experimental Results

In the following, we analyze the performance of DCB-Miner in terms of biological soundness and run time performance. For this, we first designed a benchmark competition, in which we analyze the biological accuracy and usefulness of the output of various algorithms. Secondly, we did extensive runtime analysis to verify the scalability of DCB-Miner under different parameter settings. All experiments were performed on a PC running the Linux operating system with a 1.86GHz CPU and 4 GB of main memory.

5.1 Data

In order to construct the input attributed graph for DCB-Miner and the comparison partners, we constructed data sets from yeast and human.

5.1.1 Yeast Dataset

The interaction network under consideration was extracted from the BioGRID database [73]. It integrates both PPI and GI interactions from multiple publicly available databases and datasets. Gene expression data was given by the yeast compendium dataset [39]. It reports fold changes of experiment against control in as many as 300 cDNA experiments. In order to get rid of nodes in the network that are not active under any type of condition under consideration, we removed genes that did not exhibit a significant expression pattern over the 300 experiments. Namely, we discarded genes whose ratios were to be found in a 1.5 times variance interval around the mean over all conditions. This finally amounted to

1043 variably expressed genes with 2664 interactions in the resulting network.

5.1.2 Human Dataset

Similarly, the PPI/GI network was downloaded from the BioGRID database [73]. For the expression data, we used the comprehensive human tissue expression dataset [69], which lists fold changes over 115 cDNA experiments across 35 different tissue types. In order to account for activity, we only retained variably expressed genes which were with at least 2-fold ratio variation from the mean in at least 2 samples. This preprocessing was done by the authors. As a result, the human dataset contained 3628 genes connected by 8924 interactions in the respective network.

5.2 Gene Ontology (GO) Based Evaluation

To assess the quality of the results of our algorithm, we compared it to four related publicly available, state-of-the-art algorithms. These include two integrated methods and two methods that operate on one data type (either interaction network or gene expression data) only.

5.2.1 Competition Partners

1. **SAMBA** (Statistical-Algorithmic Method for Bi-Clustering) [75] is a widely used biclustering algorithm that infers modules from expression data only. It performed comparable, if not better, in a recent comparative study of biclustering algorithms [57]. Basically, it finds sets of genes that jointly respond to changing conditions. To do this, first a weighted bipartite graph $G = (U, V, E)$ is constructed, in which U is the set of conditions, V is the set of genes and E is the set of edges such that $(u, v) \in E$ if gene v *responds* in condition u. The weights are assigned based on a statistical model that incorporate background, i.e. *null*, distribution of gene-condition edges. In the second phase, k best bicliques are found from the graph constructed in the first phase. In this context, a biclique corresponds to a functional module, whose member genes *co-respond* to a subset of experimental conditions. As common in biclustering algorithms, the result set is usually large. Therefore, in the third phase, a greedy algorithm is utilized to select non-overlapping biclusters to minimize the overlap.

2. **MCL** (Markov Clustering) [21] only considers interaction network data. It outperformed other methods of this type in the comparative study of [12]. Given a possibly weighted graph $G = (V, E)$, it first transforms the graph into a Markov graph. Then it performs a random walk, which corresponds to simulation of flow in an interaction graph by computing successive powers of the adjacency matrix of the graph. This step is known as *expansion* (multiplication) followed by *scaling* and it converges to a partition of the interaction network into dense subnetworks which are output as modules.

3. **Co-Clustering** [33] is one of the two integrated approaches we consider. Infact, the authors propose a novel distance function rather than an algorithm. The proposed distance function incorporates both similarity of gene expression profiles and network shortest path distance using a *logistic sigmoid* function. The distance function then can be plug into an arbitrary choice of a distance-based clustering algorithm. We used the proposed distance function with K-Means algorithm.

4. **Matisse** (Modular Analysis for Topology of Interactions and Similarity Sets) [80] is the most recent integrated approach available. It is a probabilistic method that finds connected subnetworks in interaction networks that exhibit high expression similarity. It starts with seed generations phase, in which a pre-specified number of highly interconnected subgraph are selected. In the optimization step, moves including addition and deletion of a node and merging and deleting of clusters are tried in order to improve an overall score. Finally, in a post-processing step, only top k significant clusters are given as the output.

For all algorithms, we used the recommended parameter sets if applicable. For Co-clustering, which is the only algorithm that requires number of modules apriori, we tried different values and selected the one with best F-value, which we describe in next section. For DCB-Miner, we set the parameters to values close to the properties of known functional modules. For this, we downloaded the yeast molecular complexes and the pathways from the Saccharomyces Genome Database (SGD) [11]. We mapped each of the modules to the network consisting of the PPI and genetic interaction network of yeast. We only focused on the modules that induce a connected subnetwork in the corresponding network; otherwise we consider the module information as incomplete and removed from the rest of the analysis. This resulted in 111 well-characterized modules. Figure 5.1 shows the density distribution

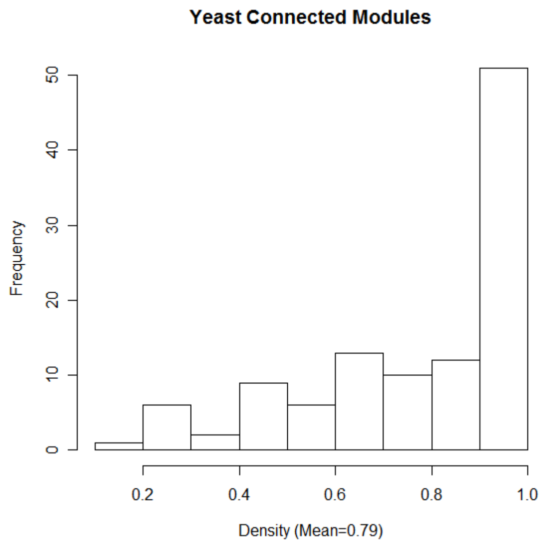


Figure 5.1: Density distribution of well characterized yeast functional modules extracted from the Saccharomyces Genome Database (SGD) [11]

of the modules. Based on the distributions of density and the number of homogenous dimensions (Figure 5.2), we set $\alpha = 0.65$, $\theta_h = 1.25$ and $\theta_{min} = 140$ (out of 300) for the yeast dataset. Contrary to yeast, there is no comprehensive true human module dataset. Therefore, we used the same density threshold, i.e. $\alpha = 0.65$. Moreover, the human expression dataset contains a high amount of missing values ($> 25\%$) which adverts to a high amount of noise. Therefore, we used a more relaxed fold change threshold of $\theta_h = 1.4$ and $\theta_{min} = 10$ (out of 115). In general, in case of existence of a partial true modulome annotation, DCB-Miner's parameters can be approximately derived from the annotated true modules. Note that, the results of DCB are post-processed as described in Section 4.3. Finally, for all algorithms, only modules of size 4 or larger are considered in the analysis.

Note that none of the related methods address the problem of finding co-active, dense subnetworks according to Definition 4. Hence, each algorithm has its own definition of a functional module, usually corresponding to subset of properties addressed by the definition of DCB. The only method that yields overlapping modules is SAMBA whereas the only one that addresses the question of finding dense subnetworks is MCL. Theoretically, MCL can also output overlapping modules, but we haven't observed this in our analysis. However, SAMBA and MCL are not integrated which makes them more prone to noise or missing data

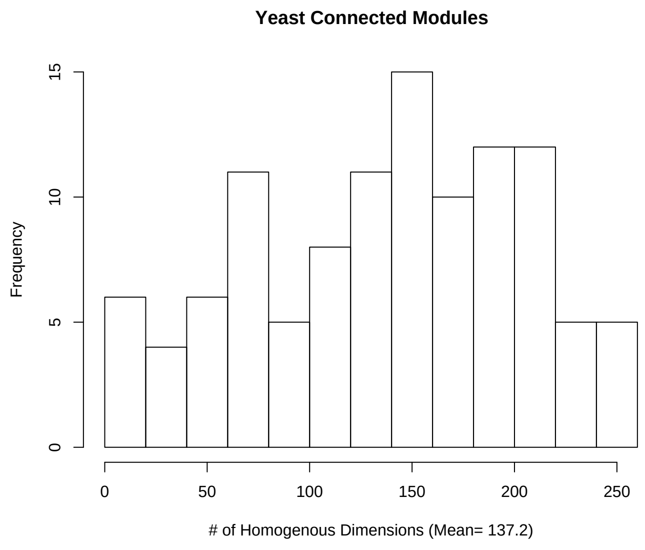


Figure 5.2: Distribution of the number of homogenous dimensions wrt. $\theta_h = 1.25$ of the well characterized yeast functional modules extracted from the Saccharomyces Genome Database (SGD) [11].

that occur in one data type only. The integrated methods, Co-Clustering and Matisse, both do not address finding dense subnetworks and they do not produce overlapping modules. Matisse usually outputs small number modules whereas Co-Clustering partitions the whole dataset. Methods that assign each gene to a cluster (MCL, Co-Clustering) may also suffer from forcibly collecting proteins into modules which, due to current amounts of noise in the data, cannot be done reliably. These are strong hints to the quantitative and qualitative superiority of our method.

5.2.2 Module Assessment

Due to the absence of comprehensive module annotations, testing for statistically significantly over-represented GO terms in a group of genes of interest is the common method used for evaluation of module inference algorithms. Basically, for each GO term T, the occurrence distribution of the term in a given module is compared against its occurrence distribution in the whole genome, which corresponds to the *null* distribution. If the term is found to be occurred statistically significantly more frequently than expected from the *null* distribution, then the module is said to be enriched with term T. This statistic can be calculated using by a chi-square test. For calculations, we used the high-throughput version of the GoMiner tool [88].

In this GO-based evaluation procedure, applied to outputs from all methods, generated from yeast (Section 5.1.1) and human (Section 5.1.2), we applied three different quantities to gauge module quality.

1. *Enrichment* is computed as the percentage of modules found that are enriched with at least one GO term (level 7 or higher, as suggested in [80]) with P-values, that were corrected for multiple hypothesis testing, below a threshold of 0.01. Roughly, this value accounts for the amount of true modules among the overall amount of predicted modules. In other words, it can be perceived as a *relative precision*, where relativity refers to the computations being done relative to a dataset.

2. *Coverage* is defined as the number of GO terms associated with an enriched cluster found by the method divided by the number of all GO terms in the dataset. This number can be perceived as as a *relative recall*.

3. The *F-Value* is a common quantity to incorporate precision and recall into a single value. Given enrichment E and coverage C, it is computed as

$$F = \frac{2EC}{E + C}.$$

   Intuitively, the *F-Value* captures the trade-off between enrichment and coverage.

4. To account for completeness, we juxtapose *absolute numbers* of inferred modules to enrichment, coverage and F-value, as these are only relative measures.

5.2.3 Results

In Figure 5.3, we have displayed the statistics defined in Section 5.2.2 that were achieved by our method and those listed in Section 5.2 on the yeast datasets. In terms of *enrichment*, all algorithms except Co-clustering perform relatively well, but only Matisse (94%) and DCB-Miner (94%) yield an enrichment over 90%. This agrees with the biological reasoning that modules that are coherent in both data types are more reliable than modules that are coherent in only one data type. Although Co-clustering is also a combined learning algorithm, it forces every gene to be an element of a cluster, which is not a realistic scenario.

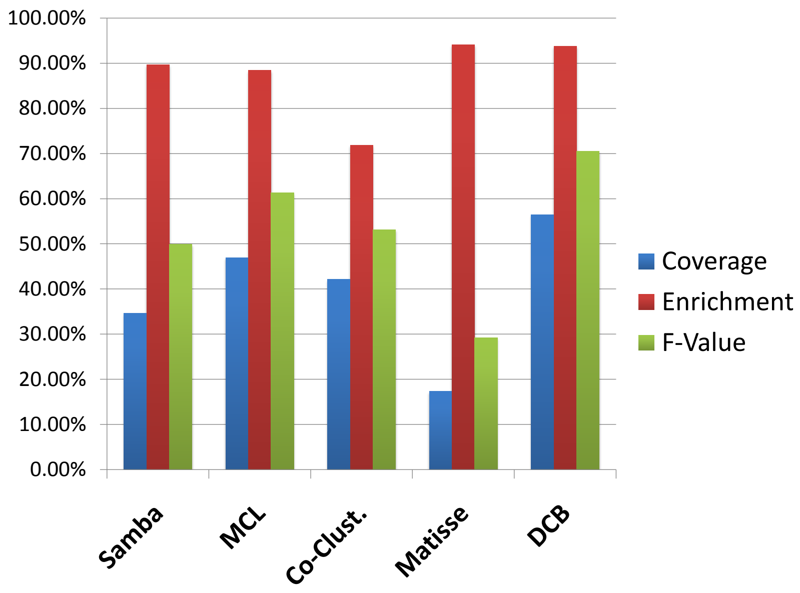


Figure 5.3: Performance of all algorithms on the yeast dataset

Moreover, the poor enrichment performance of Co-clustering also suggests that the proposed combined distance function may not be a biologically reasonable one on the instances considered. With regard to coverage, it is no surprise that MCL and Co-clustering, which force every gene to belong to a cluster, perform relatively better than Matisse and SAMBA, which output only statistically significant clusters. Although DCB-Miner does not force every gene to belong to a cluster, it achieves the best coverage over all methods, thanks to its completeness. This means that we compute modules of a larger range of functionalities in yeast than prior approaches did. According to the F-value, DCB-Miner achieves the best overall performance which gives further evidence of the superiority of the concept of active modules being encoded as densely connected biclusters.

In Figure 5.4, we have displayed the respective statistics for the human datasets. Besides from SAMBA, all methods achieve an increased coverage, thanks to the, compared to yeast, higher overall density of the PPI/GI network. Recall that SAMBA does not utilize the network data. MCL and Co-Clustering perform poorly in terms of enrichment, probably due to forcing all genes into modules. Only Matisse (93%), DCB-Miner (97%) and SAMBA (94%) yield an enrichment over 90%. The explanation for the good performance of SAMBA is that, on multiple-condition expression data only, biclustering is a highly valuable

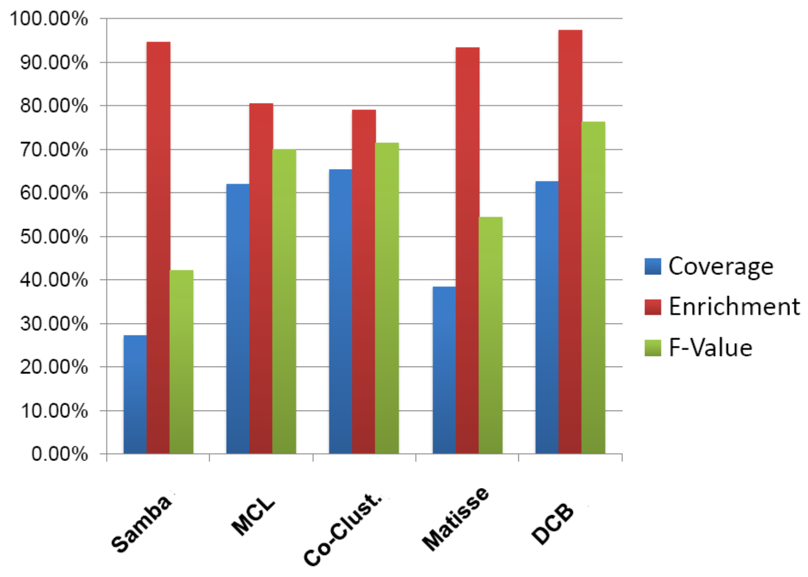


Figure 5.4: Performance of all algorithms on the human dataset

approach. As for coverage, only MCL (62%), Co-clustering (65%) and DCB-Miner (63%) yield a coverage over 60%. Overall, DCB-Miner achieves the best F-value as a result of both highest enrichment and comparable, if not better, coverage performance.

At this point, we would like to discuss an interesting finding. Similar to the analysis done in [80], we sampled random connected networks so that the size distribution is the same as the pooled size distribution of the five algorithms discussed above. We found that, even random sampling of connected networks achieves an enrichment rate as high as 85% in the yeast dataset and 89% in the human dataset. Although one would expect connected random subnetworks to be partially meaningful, the achieved enrichment rates are surprisingly comparable if not better than that performance of MCL and Co-clustering. This further validates the idea that a functional module is densely connected and co-expressed. Note that we did not compare the coverage performance of random sampling against other algorithms due the fact that the coverage of random subnetworks is highly correlated with the number of modules being sampled, i.e. one can achieve 100% coverage by sampling more and more connected subnetworks, which would not be fair against comparison partners.

An interesting question is as follows: How do the well characterized modules of the yeast break up into modules found by the algorithms? By merely looking at the overlap

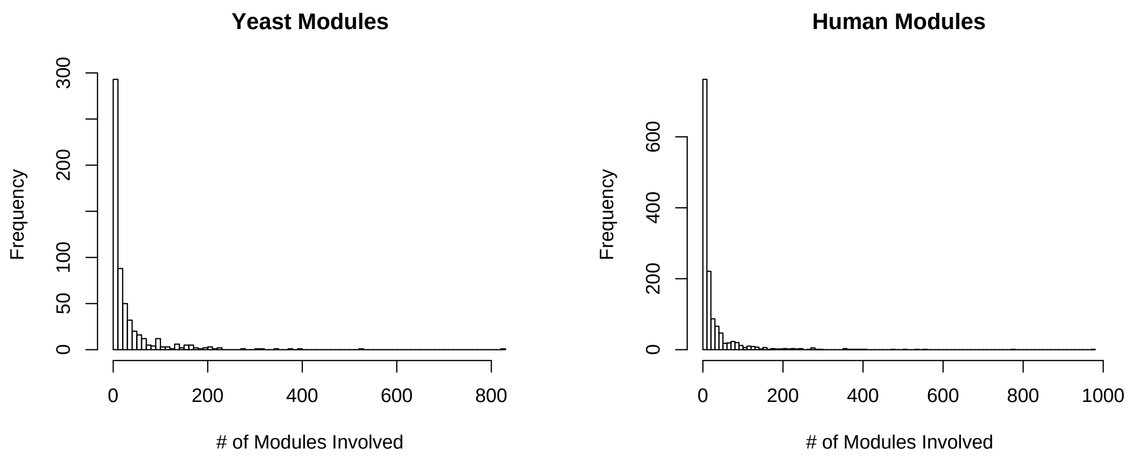


Figure 5.5: Distribution of number the modules a gene is involved in for the yeast (left) and the human (right) datasets.

between discovered modules and true modules, we found that 85 of the 111 true modules are covered by the DCB-Modules, which can be perceived as the true positives. The closest to this number is 81 achieved by MCL. On the other hand, this kind of analysis has certain drawbacks. As described in the survey work of [12], if an algorithm makes more predictions than the current level of annotation available, all novel predictions are labeled as false positives. However, such new predictions can be novel discovery rather than false predictions. Therefore, the sparsity of the current level of annotation favors algorithms with few predictions rather than algorithms that do novel predictions. Unless a more complete true module annotation is available, we believe this kind of comparison is highly biased and therefore did not do further analysis. Last but not least, in Figure 5.5 we give the distribution of the number of modules a gene is involved in. It is evident that the distributions follow a power-law distribution. Focused analysis of genes that are involved in many modules show that these are hub genes and also there is a correlation between the out degree of a gene and the number of modules the genes are involved in. This, on the other hand, suggests that proteins such as chaperons, which interact with hundreds of proteins, should be removed from the analysis as they present outliers, i.e. they interact with hundreds of genes but are known to belong to few modules.

Finally, we display some statistics on the inferred modules in Table 5.1. These demonstrate that DCB-Miner clearly outperforms the other algorithms with respect to absolute

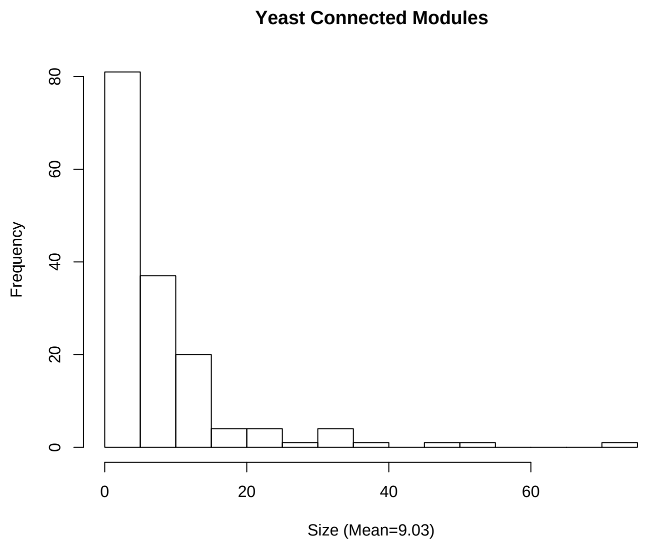


Figure 5.6: Size distribution of well characterized yeast functional modules extracted from the Saccharomyces Genome Database (SGD) [11].

numbers of highly reliable modules of various functionalities (for additional information on size distributions of the found DCB-Miners see the Figure 5.7). As we infer the entirety of modulome, numbers achieved by our method also reflect estimates on the sizes of both yeast and human modulomes. Actual guesses on the numbers of complexes in yeast resp. human are 800 resp. 3000 [92]. It can safely be assumed that respective numbers of modules, which are not necessarily complexes, are much greater. As none of the other methods addresses the issue of inferring the complete modulomes, it comes as no surprise that, compared to these estimates, they fail to predict the modulomes' sizes in terms of orders of magnitude. Last but not least, although DCB-Miner algorithm outputs many modules, the most probable use case for a molecular biologist consists of focused analysis of a set of genes, the modules they are involved in and their interplay. The user can always select the modules in which the genes of interest are involved in and do focused analysis of those modules. As a result of being a complete algorithm, DCB-Miner provides the complete solution to the user as well as the option of focused analysis. Finally, modules found by DCB-Miner and Co-clustering seem to match the average size of the true connected modules in yeast as shown in Figure 5.6. MCL outputs modules of very small size ,whereas SAMBA outputs modules of very large size, which makes analysis very hard. Due to lack of comprehensive

Table 5.1: Module Statistics.

Competition Partners						
	Yeast			Human		
Method	**# of Mod.**	**Density**	**Avg. Size**	**# of Mod.**	**Density**	**Avg. Size**
Samba	135	.02	25.06	129	.01	48.94
MCL	95	.44	7.29	312	.35	5.94
Matisse	17	.31	21.17	76	.30	17.94
Co-Clustering	103	.06	9.57	271	.01	13.12
DCB	2276	.39	8.05	5979	.46	7.12

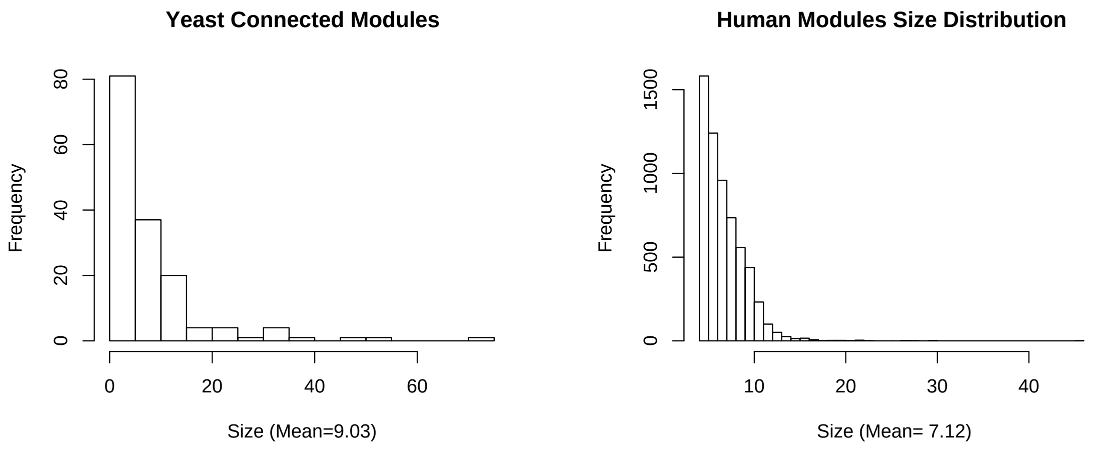


Figure 5.7: Distribution of the sizes of DCBs found in the yeast (left) and the human (right) datasets.

human module annotation, it is hard to make comments on the sizes of the human modules at this point. Still, one needs to be careful as the majority of the modules in the yeast true connected module dataset are complexes rather than pathways. This introduces a bias towards smaller modules. In this respect, Matisse seems to best approximate the true module sizes, especially those of the pathways'. However, we believe that with more complete interaction data and gene expression data of better quality, modules found by our algorithm will be able to match the true size distribution of the real modules.

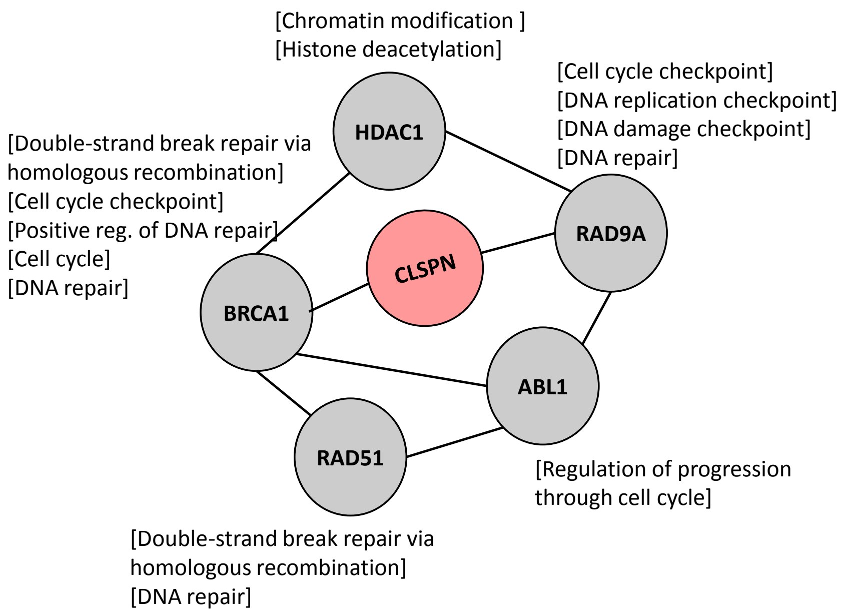

[Chromatin modification]
[Histone deacetylation]
HDAC1
[Cell cycle checkpoint]
[DNA replication checkpoint]
[DNA damage checkpoint]
[DNA repair]
RAD9A
[Double-strand break repair via homologous recombination]
[Cell cycle checkpoint]
[Positive reg. of DNA repair]
[Cell cycle]
[DNA repair]
BRCA1
CLSPN
ABL1
RAD51
[Regulation of progression through cell cycle]
[Double-strand break repair via homologous recombination]
[DNA repair]

Figure 5.8: Annotation of CLSPN gene via other annotated genes in the same enriched module

5.3 Prediction of Novel Annotation

In this section, we illustrate how the DCB-Miner algorithm can be used for novel annotation prediction. We present two case studies where previously crudely annotated genes can be annotated more specifically based on the *guilt-by-association* principle. Basically, we find modules that are enriched with some GO terms yet contain genes of unknown (or partially unknown) function [66]. We then assign the functions implied by the enriched GO terms as the function of the gene of unknown function. Note that this approach works with only GO terms of type localization and biological process but not with type molecular function. This is because elements of a functional module is expected to be involved in a common biological process and also to co-localize. However, within a biological process each protein usually has a distinct molecular function. We illustrate the *guilt-by-association* in the following examples. Two modules (one in human and one in yeast) that are enriched with cell cycle regulation and resp. cell division related GO terms were chosen as examples. The human module (Figure 5.8) consists of six genes including CLSPN, a recently characterized yet crudely annotated gene, and extensively annotated genes such as BRCA1. The yeast module

(Figure 5.9) has eight genes including LSB1, a gene of unknown function.

Cell cycle is a tightly regulated process that depends on a host of parameters. Many cancers are due to disruption in the cell cycle control. DNA repair is a crucial process during cell cycle. Some of the cancer types, such as breast cancer, are associated with disruption of the DNA repair mechanism for single stranded and double stranded DNA breaks [22]. Claspin (CLSPN) is a recently characterized protein to function in repairing DNA single stranded or double stranded breaks. Claspin interacts with the chromatin early on in the replication process as well as the replication machinery [13]. At the site of damage, replication is halted and Claspin disassociates to interact with Rad9, a member of the 9-1-1 complex [93], and promotes interaction with HDAC1 [15], BRCA1 [50] or CHEK1. CHEK1 stalls the cell cycle at the S phase until nucleosome is remodeled via HDAC1 and DNA is repaired via BRCA1 and RAD51 [72]. In the case of severe DNA damage, apoptotic pathways are induced and claspin is ultimately degraded by ubiquitination [64]. Based on the enriched GO terms, we predict process annotations: DNA replication checkpoint (GO:0000076), regulation of DNA replication initiation (GO:0030174), regulation of DNA repair (GO:0006282), and DNA damage checkpoint (GO:0000077). We further predict the cellular component annotation to be: chromosome (GO:0005694).

Yeast gives rise to daughter cells by budding and subsequent cytokinesis to separate the newly budded cell from the mother. Cytokinesis depends heavily on actin assembly and disassembly. Actin or cytoskeleton abnormalities have been observed in hematopoietic cells of Wiskott-Aldrich syndrome patients [70]. LAS17 is the yeast homolog of the main factor (WASP) associated with this disease. The process of actin assembly requires nucleation factors such as LAS17, HOF1, VRP1, and RVS167. LAS17 associates with VRP1 and RVS167 via its SH3 domains [51]. VRP1 then recruits HOF1 to the complex to facilitate cytokinesis [59]. The role of ABP1 is thought to be recycling nucleation factors for further assembly as well as attenuating the rate of assembly to ensure proper timing [18]. NCP1, a protein involved in sterol synthesis, is also suggested to play a role in bud site selection and establishment of cell polarity [77]. LSB1, which lacks molecular function and biological process annotation, is known to have SH3 domains that interact with LAS17 [51] and therefore it is likely to be a novel nucleation factor participating in actin assembly process. In this respect, our prediction for its biological process being actin polymerization and/or depolymerization (GO:0008154), actin filament organization (GO:0007015), and establishment of cell polarity (GO:0030010) agrees with the current literature. Furthermore, we are able to annotate

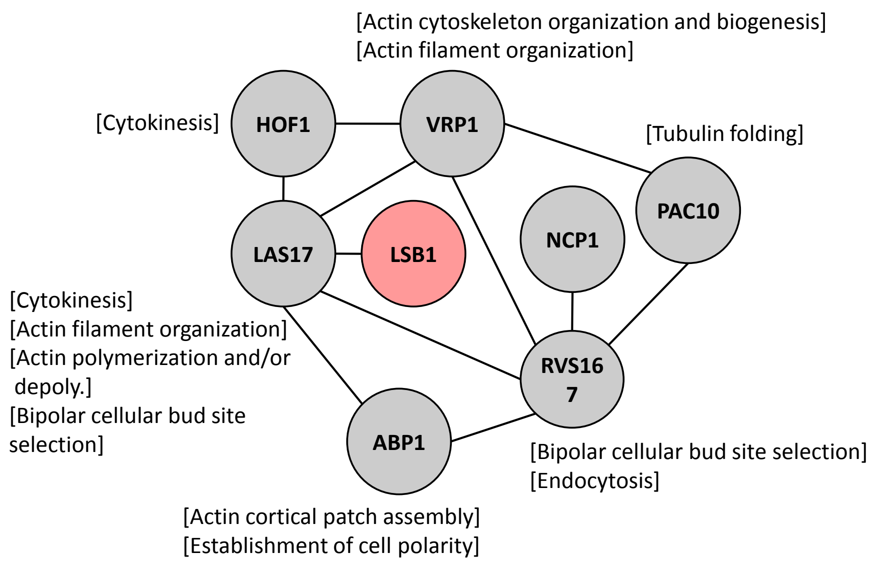

[Actin cytoskeleton organization and biogenesis]
[Actin filament organization]
[Cytokinesis]
HOF1
VRP1
[Tubulin folding]
PAC10
NCP1
LAS17
LSB1
[Cytokinesis]
[Actin filament organization]
[Actin polymerization and/or depoly.]
[Bipolar cellular bud site selection]
RVS167
ABP1
[Bipolar cellular bud site selection]
[Endocytosis]
[Actin cortical patch assembly]
[Establishment of cell polarity]

Figure 5.9: Annotation of LSP1 gene via other annotated genes in the same enriched module

more specifically its cellular component to be actin cortical patch (GO:0030479) and cortical actin cytoskeleton (GO:0030864). Interestingly, our results suggested that VRP1 and RVS167 interact with PAC10 (YGR078C), which is a chaperone for microtubule proteins [49]. It is likely that PAC10 may have a more general function in assisting proper folding of proteins involved in cytoskeleton biogenesis.

Overall, DCB-Miner predicted novel GO annotations (biological process and cellular localization at level 7 or higher) for 87 of the 188 un-annotated genes in the yeast dataset and for the 31 out of the 84 un-annotated genes in the human dataset. to our best knowledge, we have not observed any of our predictions to be conflicting with existing literature. This is because an extensive analysis of our prediction requires a considerable amount of literature survey , which is out of the scope of this thesis.

5.4 Runtime Experiments

In this section we analyze the runtime performance of DCB-Miner algorithm. We use the yeast data set described in Section 5.1 for this purpose. Recall that DCB-Miner has three parameters, which are the minimum density α, homogeneity threshold θ_h and the minimum

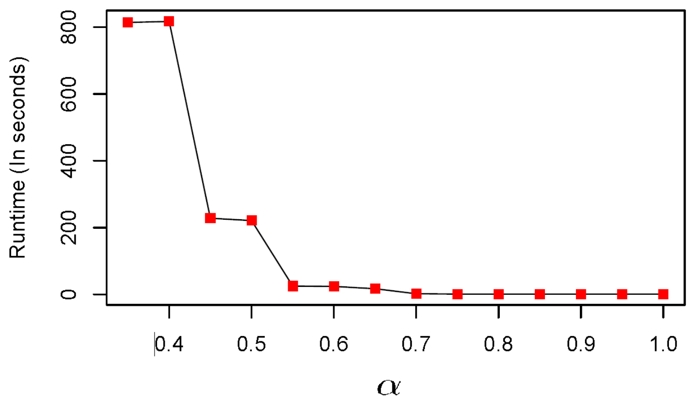


Figure 5.10: α vs runtime

number of homogenous dimensions θ_{min}. In the following, we analyze how the performance of DCB-Miner changes with regards to the changes in each of the three parameters. Our major observation is that the performance of DCB-Miner is more susceptible to the density parameter than other parameters. This is not surprising because a density value below $\frac{1}{2}$ requires the execution of the *Merge* phase which increases the runtime significantly. This finding also implies that the majority of the search space pruning capability comes from the graph based constraints, i.e. connectedness and the density constraints. Even in the absence of pruning capabilities of the homogeneity constraint , i.e. setting $\theta_d = 0$ or $\theta_h = 3$, the algorithm is still able to finish in a reasonable amount of time thanks to the pruning provided by the graph based constraints.

Figure 5.10 shows the runtime results in response to changing density threshold with homogeneity threshold fixed to $\theta_h = 1.25$ and minimum number of dimensions fixed to $\theta_{min} = 140$. DCB-Miner is quite efficient if density threshold is close to or greater than the average density of the true biological modules, which is 0.79 (See Figure 5.1). There are two significant increase points. The first one is at $\alpha = \frac{1}{2}$, which is due to the fact every connected subgraph of size 4 or less has density greater than $\frac{1}{2}$. Therefore, an increase in runtime is reasonable. The second significant increase occurs when $\alpha < 0.5$. This is expected because the loose anti-monotonicity property of density constraint does not hold below $\alpha < \frac{1}{2}$ and the expensive *Merge* phase is needs to be called in that case. In addition to this, as the density threshold is lowered, combinatorially many more subgraphs satisfy the density threshold. Similar to typical pattern mining algorithms, DCB-Miner is also an

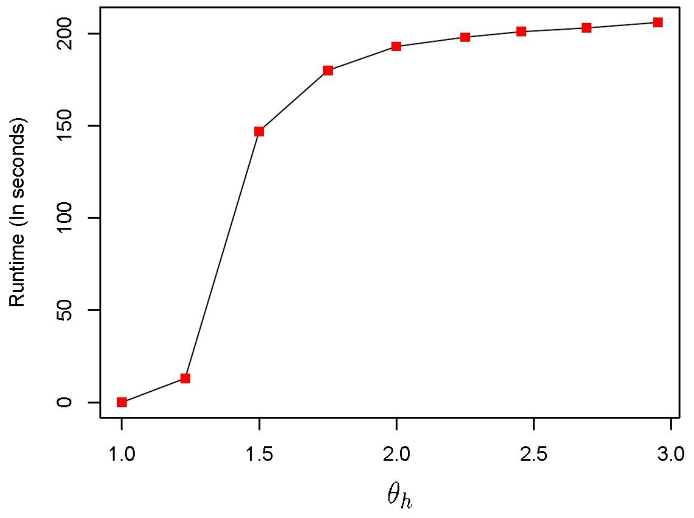


Figure 5.11: θ_h vs runtime

output dependent algorithm meaning that the more patterns exist in the data set, the higher the run time. Finally, note that $\alpha = \frac{1}{3}$ is not a realistic density threshold because every connected subgraph of size 6 or less satisfy this threshold. This means that this density threshold would fail to separate random connected subnetworks from true modules. Still, even in the worst case of $\alpha = 0.35$, DCB-Miner takes less than 15 minutes, which is a reasonable amount of time.

Figure 5.11 shows the runtime results with regard to the changes in the homogeneity, i.e. fold change range, threshold with density fixed to $\alpha = 0.65$ and minimum number of dimensions fixed to $\theta_{min} = 140$. We see a drastic increase at $\theta_h = 1.5$. This can be explained by the fact that majority of fold change difference values are below 1.5. After 1.5, increasing the homogeneity range does not change the runtime significantly because almost all genes are within 1.5 fold change neighborhood of each other.

Figure 5.12 shows the runtime against the number of homogenous dimensions with minimum density fixed to $\alpha = 0.65$ and fold change range threshold fixed to $\theta_{min} = 1.25$. As expected, runtime decreases as the minimum number of homogenous dimensions increase. If minimum number of homogenous dimensions is set close to average number of homogenous dimensions of annotated complexes, which we found to be 137, performance is quite satisfactory as the runtime is only 18 seconds. Even if homogeneity threshold is decreased to 0, we see that DCB-Miner takes only 108 seconds. This shows that even in the absence

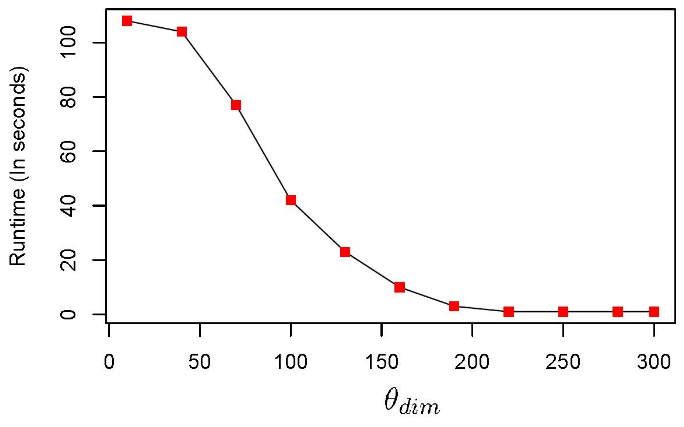


Figure 5.12: θ_{dim} vs runtime

of prunning capabilities of attribute data, breadth-first approach of DCB-Miner is able to mine dense subgraphs in a reasonable amount of time thanks to the loose anti-monotonicity of density constraint.

The above runtime results contain the *CheckMaximality* procedure. Although the designed index structure is very efficient and takes significantly lower time than the core phases of the algorithm, i.e. the *Expand-by-one* and the *Merge* phases, we observed that maximality is almost never a problem, i.e. only a small fraction of DCBs were found to be non-maximal during *CheckMaximality* phase. This is especially the case for larger values of density threshold, i.e. the higher the α is the higher the percentage of permutations of a DCB that satisfy the α-strong connectedness property. In this respect, removing *Check-Maximality* phase can further increase the runtime at a cost of some redundancy in the result set. In addition to this, the above runtime results do not include the optional post processing step which is considerably slower than DCB-Miner. However, we observed that, even in the absence of post-processing the enrichment and coverage scores are comparable if not better than the results obtained with post-processing. In this respect, post-processing should be considered as a summarization phase for making it easy to analyze the found modules. It comes with a cost of runtime and possible decrease in quality, which is due to the relaxation of the constraints.

Finally, we would like to once more emphasize that the parameters for DCB-Miner can be derived by doing an exploratory analysis of distributions of the parameters with respect to a partial module annotations of the organism. We followed this approach in our experiments and this resulted in 18 and 8 seconds of runtime for the yeast and the human datasets respectively. This is quite efficient considering the significance of the output of the DCB-Miner, which is the modulome of the organism being analyzed.

Chapter 6

Conclusion and Future Work

6.1 Contributions

In this thesis, we proposed a novel method that can computationally infer the entirety of an organism's modules that are active under a variety of cellular conditions specified by a set of gene expression experiments. This was done by defining a module as a set of genes that are co-expressed and show significant changes in expression under a sufficiently large number of experimental conditions on one hand and induce a dense subnetwork in the PPI/GI network on the other hand. While the biological literature shows that this definition of a module is most natural, the corresponding search problem has not been tackled before due to its computational hardness. We solve the problem by a carefully designed algorithmic search strategy. We demonstrated its effectiveness by finding large numbers of high quality modules in both yeast and human, thereby outperforming a variety of prominent related approaches. Moreover, we presented novel functional annotations of proteins in yeast and human, as predicted by our method.

6.2 Future work

We finally would like to point out that there is still a lot of room for improvements and future work. Below, we summarize some possible future extensions.

- An issue originating from the molecular biology is the still substantial rate of not yet documented protein-protein and genetic interactions in various organisms which can result in very sparse networks. In our experience, our algorithm will yield suboptimal results on such instances. However, this issue is only temporary in nature and presumably will be resolved in the near future at least for the model organisms.

- Another issue is that the homogeneity constraint used in the DCB definition is a rough measure for co-expression and does not reflect the actual state of the art when inferring modules on the basis of gene expression data. The integration of more suitable co-expression models promises to further improve the biological validity of the DCB model. For example, Absolute Pearson Correlation Distance can be used as a homogeneity constraint. An example constraint can be as follows: *Maximum Absolute Pearson Correlation Distance between elements of a functional module must be smaller than* 0.5. However, Pearson Correlation Distance is neither anti-monotone nor loose anti-monotone and hence its tractability remains a challenging yet promising open problem in the context of biclustering. Therefore, in order to use Pearson Correlation Distance, one must stick to the full space rather than subspaces. This would be useful if the dimensionality of the expression dataset is low. Note finally that, the longest common increasing subsequence (LCIS) problem has recently found applications in the biclustering area due to its biological soundness [25]. However, we observed that for a set of genes, contrary to the homogeneity constraint used in this thesis, the LCIS is not unique and one needs to keep all possible LCISs while doing the breadth first search. We found this to be a computationally expensive requirement especially in the case of high dimensional datasets. However, the LCIS based formulation may be feasible with different approaches and hence is an open yet a very promising research direction.

- DCB-Miner algorithm proposes a generic solution for combining attribute and relational data. Homogeneity constraints could also be based on other attribute data sources such as literature co-occurrence and/or phenotypic profiles. As long as the homogeneity constraint is anti-monotone, it can be plugged in to DCB-Miner. Similarly, other biological network data can be used instead of (or in combination with) PPI and/or GI networks. For example, functional modules are shown to induce densely connected subnetworks in regulatory networks as well [92]. Metabolic and phosphorylation networks are also worthwhile to investigate.

- We have temporarily solved the high overlap problem in a heuristic post-processing step (See Section 4.3). Although, the post-processing step is able to reduce the number of modules from 7837 and 14274 to 2125 and 4974 in yeast and human respectively, it comes with the cost of additional runtime and sacrifice in quality of the found DCBs. For example, the enrichment rates for the yeast dataset are 96% and 96% before and after post-processing respectively. This can be explained by the fact that homogeneity and density constraints are relaxed in the post-processing phase. Currently, we are working on a statistical method for the post-processing step, which is more efficient and can also calculate a quality score for each DCB.

- Another interesting research direction is extending the model to weighted graphs. This would be beneficial in a lot of cases. For example, one of the ways to deal with the noise in interaction networks is assigning confidence weights to edges based on the number of publications in which the interaction was experimentally observed. In such a context, finding dense regions would correspond to finding highly interacting gene groups with high confidence.

- Finally, extending the model into multiple graph mining would also be very useful. For example, integration of multiple microarray data sets across many platforms for integrated analysis poses challenging problems as discussed in Section 2.1. One way to deal with this problem is building a co-expression graphs for each dataset and mining these graphs jointly [38, 37, 84, 56]. Our model is easily expandable to handle such cases.

Bibliography

[1] James Abello, Mauricio G. C. Resende, and Sandra Sudarsky. Massive quasi-clique detection. In *LATIN '02: Proceedings of the 5th Latin American Symposium on Theoretical Informatics*, pages 598–612, London, UK, 2002. Springer-Verlag.

[2] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery*, 11(1):5–33, 2005.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94: Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[4] Reka Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118(21):4947–4957, 2005.

[5] Sitaram Asur, Duygu Ucar, and Srinivasan Parthasarathy. An ensemble framework for clustering protein protein interaction networks. *Bioinformatics*, 23(13):i29–40, 2007.

[6] Gary D. Bader and Christopher WV Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4:2, 2003.

[7] Albert-Laszlo Barabasi and Zoltan N. Oltvai. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, February 2004.

[8] Celine Becquet, Sylvain Blachon, Baptiste Jeudy, Jean-Francois Boulicaut, and Olivier Gandrillon. Strong-association-rule mining for large-scale gene-expression data analysis: a case study on human sage data. *Genome Biology*, 3(12):research0067.1–research0067.16, 2002.

[9] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB '02: Proceedings of the 6th Annual International Conference on Computational Molecular Biology*, pages 49–57, New York, NY, USA, 2002. ACM.

[10] Francesco Bonchi and Claudio Lucchese. Pushing tougher constraints in frequent pattern mining. In *PAKDD '05: Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 114–124, 2005.

[11] David Botstein, Steven A. Chervitz, and J. Michael Cherry. Yeast as a model organism. *Science*, 277(5330):1259–60, 1997.

[12] Sylvain Brohee and Jacques van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7:488, November 2006.

[13] Marc Brondelloa, Bernard Ducommunb, Anne Fernandezc, and Ned J. Lamb. Linking pcna-dependent replication and atr by human claspin. *Biochemical and Biophysical Research Communications*, 354(4):1028–1033, 2007.

[14] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, Guojie Li, and Runsheng Chen. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Research*, 31(9):2443–2450, 2003.

[15] Richard L. Cai, Yan Yan-Neale, Maria A. Cueto, Hong Xu, and Dalia Cohen. HDAC1, a Histone Deacetylase, Forms a Complex with Hus1 and Rad9, Two G2/M Checkpoint Rad Proteins. *Journal of Biological Chemistry*, 275(36):27909–27916, 2000.

[16] Guang Chen, Shane Jensen, and Christian Stoeckert. Clustering of genes into regulons using integrated modeling-cogrim. *Genome Biology*, 8(1):R4, 2007.

[17] Yizong Cheng and George M. Church. Biclustering of expression data. In *ISMB '00: Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press, 2000.

[18] Jessica L. D'Agostino and Bruce L. Goode. Dissection of Arp2/3 Complex Actin Nucleation Mechanism and Distinct Roles for Its Nucleation-Promoting Factors in Saccharomyces cerevisiae. *Genetics*, 171(1):35–47, 2005.

[19] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *KDD '04: Proceedings of the 10th International Conference on Knowledge Discovery in Data mining*, pages 551–556. ACM, 2004.

[20] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[21] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 2002.

[22] Hannele Erkko, Katri Pylks, Sanna-Maria Karppinen, and Robert Winqvist. Germline alterations in the clspn gene in breast cancer families. *Cancer Letters*, 261(1):93–97, 2007.

[23] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD '96: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[24] Christos Faloutsos and Vasileios Megalooikonomou. On data mining, compression, and kolmogorov complexity. *Data Mining and Knowledge Discovery*, 15(1):3–20, 2007.

[25] Byron J. Gao, Obi L. Griffith, Martin Ester, and Steven J. M. Jones. Discovering significant opsm subspace clusters in massive gene expression data. In *KDD '06: Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining*, pages 922–928, New York, NY, USA, 2006. ACM.

[26] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[27] Audrey Gasch and Michael Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3(11):research0059.1–research0059.22, 2002.

[28] H. Ge, Z. Liu, G. M. Church, and M. Vidal. Correlation between transcriptome and interactome mapping data from saccharomyces cerevisiae. *Nature Genetics*, 29(4):482–486, December 2001.

[29] Hui Ge, Albertha. J.M. Walhout, and Marc Vidal. Integrating 'omic' information: a bridge between genomics and systems biology. *Trends in Genetics*, 19(10):551–560, October 2003.

[30] Andrei Grigoriev. A relationship between gene expression and protein interactions on the proteome scale: analysis of the bacteriophage t7 and the yeast saccharomyces cerevisiae. *Nucleic Acids Research*, 29(17):3513–3519, 2001.

[31] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, September 2000.

[32] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00: Proceedings of the 19th International Conference on Management of Data*, pages 1–12, New York, NY, USA, 2000. ACM.

[33] Daniel Hanisch, Alexander Zien, Ralf Zimmer, and Thomas Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(Suppl. 1):145–154, 2002.

[34] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(4-6):175–181, 2000.

[35] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, page 627, Washington, DC, USA, 1996. IEEE Computer Society.

[36] Fereydoun Hormozdiari, Petra Berenbrink, Natasa Przulj, and Süleyman Cenk Sahinalp. Not all scale free networks are born equal: The role of the seed graph in ppi network emulation. In *Systems Biology and Computational Proteomics*, pages 1–13, 2006.

[37] Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21:i213–221, 2005.

[38] Yu Huang, Haifeng Li, Haiyan Hu, Xifeng Yan, Michael S. Waterman, Haiyan Huang, and Xianghong Jasmine Zhou. Systematic discovery of functional modules and context-specific functional annotation of human genome. *Bioinformatics*, 23(13):i222–229, 2007.

[39] Timothy R. Hughes, Matthew J. Marton, Allan R. Jones, Christopher J. Roberts, Roland Stoughton, Christopher D. Armour, Holly A. Bennett, Ernest Coffey, Hongyue Dai, Yudong D. He, Matthew J. Kidd, Amy M. King, Michael R. Meyer, David Slade, Pek Y. Lum, Sergey B. Stepaniants, Daniel D. Shoemaker, Daniel Gachotte, Kalpana Chakraburtty, Julian Simon, Martin Bard, and Stephen H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, July 2000.

[40] Trey Ideker, Owen Ozier, Benno Schwikowski, and Andrew F. Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, 18(Suppl.1):233–240, 2002.

[41] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23, London, UK, 2000. Springer-Verlag.

[42] Hawoong Jeong, Sean P. Mason, Albert-Laszlo Barabasi, and Zoltan N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[43] Liping Ji and Kian-Lee Tan. Mining gene expression data for positive and negative co-regulated gene clusters. *Bioinformatics*, 20(16):2711–2718, 2004.

[44] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[45] Jyotsna Kasturi and Raj Acharya. Clustering of diverse genomic data using information fusion. *Bioinformatics*, 21(4):423–429, 2005.

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

[47] Nevan J. Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P. Tikuisis, Thanuja Punna, Josãc M. Peregrãn Alvarez, Michael Shales, Xin Zhang, Michael Davey, Mark D. Robinson, Alberto Paccanaro, James E. Bray, Anthony Sheung, Bryan Beattie, Dawn P. Richards, Veronica Canadien, Atanas Lalev, Frank Mena, Peter Wong, Andrei Starostine, Myra M. Canete, James Vlasblom, Samuel Wu, Chris Orsi, Sean R. Collins, Shamanta Chandran, Robin Haw, Jennifer J. Rilstone, Kiran Gandi, Natalie J. Thompson, Gabe Musso, Peter St Onge, Shaun Ghanny, Mandy H. Y. Lam, Gareth Butland, Amin M. Altaf-Ul, Shigehiko Kanaya, Ali Shilatifard, Erin O'Shea, Jonathan S. Weissman, James C. Ingles, Timothy R. Hughes, John Parkinson, Mark Gerstein, Shoshana J. Wodak, Andrew Emili, and Jack F. Greenblatt. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084), March 2006.

[48] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 313–320, Washington, DC, USA, 2001. IEEE Computer Society.

[49] Soni Lacefield and Frank Solomon. A novel step in beta-tubulin folding is important for heterodimer formation in saccharomyces cerevisiae. *Genetics*, 165(2):531–541, October 2003.

[50] Shiaw-Yih Lin, Kaiyi Li, Grant S. Stewart, and Stephen J. Elledge. Human Claspin works with BRCA1 to both positively and negatively regulate cell proliferation. *Proceedings of the National Academy of Sciences*, 101(17):6484–6489, 2004.

[51] Ammar Madania, Pascal Dumoulin, Sandrine Grava, Hiroko Kitamoto, Claudia Scharer-Brodbeck, Alexandre Soulard, Violaine Moreau, and Barbara Winsor. The Saccharomyces cerevisiae Homologue of Human Wiskott-Aldrich Syndrome Protein Las17p Interacts with the Arp2/3 Complex. *Molecular Biology of the Cell*, 10(10):3521–3538, 1999.

[52] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computuational Biology and Bioinformatics*, 1(1):24–45, 2004.

[53] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained association rules. In *SIGMOD '98: Proceedings of the 17th International Conference on Management of Data*, pages 13–24, 1998.

[54] David Page and Mark Craven. Biological applications of multi-relational data mining. *SIGKDD Explorations Newsletters*, 5(1):69–79, 2003.

[55] Jian Pei, Daxin Jiang, and Aidong Zhang. Mining cross-graph quasi-cliques in gene expression and protein interaction data. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 353–354, Washington, DC, USA, 2005. IEEE Computer Society.

[56] Jian Pei, Daxin Jiang, and Aidong Zhang. On mining cross-graph quasi-cliques. In *KDD '05: Proceedings of the 11th International Conference on Knowledge Discovery in Data mining*, pages 228–238, New York, NY, USA, 2005. ACM.

[57] Amela Prelic, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Buhlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, page btl060, 2006.

[58] N. Przulj, D. G. Corneil, and I. Jurisica. Efficient estimation of graphlet frequency distributions in protein-protein interaction networks. *Bioinformatics*, 22(8):974–980, 2006.

[59] Gang Ren, Juan Wang, Ross Brinkworth, Barbara Winsor, Bostjan Kobe, and Alan L. Munn. Verprolin cytokinesis function mediated by the hof one trap domain. *Traffic*, 6(7):575–593, 2005.

[60] Alexander Schliep, Christine Steinhoff, and Alexander Schönhuth. Robust inference of groups in gene expression time-courses using mixtures of hmms. *Bioinformatics*, 20(1):283–289, 2004.

[61] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, June 2003.

[62] E. Segal, H. Wang, and D. Koller. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, 19:i264–272, 2003.

[63] E. Segal, R. Yelensky, and D. Koller. Genome-wide discovery of transcriptional modules from DNA sequence and gene expression. *Bioinformatics*, 19:i273–282, 2003.

[64] J. I. Semple, V. A. J. Smits, J. R. Fernaud, I. Mamely, and R. Freire. Cleavage and degradation of claspin during apoptosis by caspases and the proteasome. *Cell Death and Differentiation*, 14(8):14331442.

[65] Roded Sharan, Trey Ideker, Brian P. Kelley, Ron Shamir, and Richard M. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. In *RECOMB '04: Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology*, pages 282–289, New York, NY, USA, 2004. ACM.

[66] Roded Sharan, Igor Ulitsky, and Ron Shamir. Network-based prediction of protein function. *Molecular Systems Biology*, 3, 2007.

[67] Qizheng Sheng, Yves Moreau, Frank De Smet, and Kathleen Marchaland Bart De Moor. *Data Analysis and Visualization in Genomics and Proteomics*, chapter Advances in Cluster Analysis of Microarray Data, pages 153–173. John Wiley and Sons, 2005.

[68] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[69] Radha Shyamsundar, Young Kim, John Higgins, Kelli Montgomery, Michelle Jorden, Anand Sethuraman, Matt van de Rijn, David Botstein, Patrick Brown, and Jonathan Pollack. A dna microarray survey of gene expression in normal human tissues. *Genome Biology*, 6(3):R22, 2005.

[70] Scott B. Snapper and Fred S. Rosen. The wiskott-aldrich syndrome protein (wasp): roles in signaling and cytoskeletal organization. *Annual Review of Immunology*, 17:905–929, 1999.

[71] Victor Spirin and Leonid A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123–12128, 2003.

[72] Claus S. Srensen, Lasse T. Hansen, Jaroslaw Dziegielewski, Randi G. Syljuåsen, Cecilia Lundin, Jiri Bartek, and Thomas Helleday. The cell-cycle checkpoint kinase chk1 is required for mammalian homologous recombination repair. *Nature Cell Biology*, 7(2):195–201, January 2005.

[73] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. BioGRID: a general repository for interaction datasets. *Nucleic Acids Research*, 34(Database issue):535–539, 2006.

[74] Pablo Tamayo, Donna Slonim, Jill Mesirov, Qing Zhu, Sutisak Kitareewan, Ethan Dmitrovsky, Eric S. Lander, and Todd R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proceedings of the National Academy of Sciences*, 96(6):2907–2912, 1999.

[75] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18:S136–144, 2002.

[76] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281–285, July 1999.

[77] Christopher Tiedje, Daniel G. Holland, Ursula Just, and Thomas Hofken. Proteins involved in sterol synthesis interact with Ste20 and regulate cell polarity. *Journal of Cell Science*, 120(20):3613–3624, 2007.

[78] Amy H. Y. Tong, Guillaume Lesage, Gary D. Bader, Huiming Ding, Hong Xu, Xiaofeng Xin, James Young, Gabriel F. Berriz, Renee L. Brost, Michael Chang, Yiqun Chen, Xin Cheng, Gordon Chua, Helena Friesen, Debra S. Goldberg, Jennifer Haynes, Christine Humphries, Grace He, Shamiza Hussein, Lizhu Ke, Nevan Krogan, Zhijian Li, Joshua N. Levinson, Hong Lu, Patrice Menard, Christella Munyana, Ainslie B. Parsons, Owen Ryan, Raffi Tonikian, Tania Roberts, Anne-Marie Sdicu, Jesse Shapiro, Bilal Sheikh, Bernhard Suter, Sharyl L. Wong, Lan V. Zhang, Hongwei Zhu, Christopher G. Burd, Sean Munro, Chris Sander, Jasper Rine, Jack Greenblatt, Matthias Peter, Anthony Bretscher, Graham Bell, Frederick P. Roth, Grant W. Brown, Brenda Andrews, Howard Bussey, and Charles Boone. Global mapping of the yeast genetic interaction network. *Science*, 303(5659):808–813, February 2004.

[79] O. G. Troyanskaya, K. Dolinski, A. B. Owen, R. B. Altman, and D. Botstein. A bayesian framework for combining heterogeneous data sources for gene function prediction (in saccharomyces cerevisiae). *Procedings of the National Academy of Sciences*, 100(14):8348–8353, July 2003.

[80] Igor Ulitsky and Ron Shamir. Identification of functional modules using network topology and high-throughput data. *BMC Systems Biology*, 1(8), 2008.

[81] C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417(6887):399–403, May 2002.

[82] Jianyong Wang, Zhiping Zeng, and Lizhu Zhou. Clan: An algorithm for mining closed cliques from large dense graph databases. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 73, Washington, DC, USA, 2006. IEEE Computer Society.

[83] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. On compressing frequent patterns. *Data and Knowledge Engineering*, 60(1):5–29, 2007.

[84] Xifeng Yan, Michael R. Mehan, Yu Huang, Michael S. Waterman, Philip S. Yu, and Xianghong Jasmine Zhou. A graph-based approach to systematically reconstruct human transcriptional regulatory modules. *Bioinformatics*, 23(13):i577–586, 2007.

[85] Xifeng Yan, X. Jasmine Zhou, and Jiawei Han. Mining closed relational graphs with connectivity constraints. In *KDD '05: Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 324–333, New York, NY, USA, 2005. ACM.

[86] Chengyong Yang, Erliang Zeng, Tao Li, and Giri Narasimhan. Clustering genes using gene expression and text literature data. In *CSB '05: Proceedings of the 4th IEEE Computational Systems Bioinformatics Conference*, pages 329–340, Washington, DC, USA, 2005. IEEE Computer Society.

[87] K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, and W. L. Ruzzo. Model-based clustering and data transformations for gene expression data. *Bioinformatics*, 17(10):977–987, 2001.

[88] Barry Zeeberg, Haiying Qin, Sudarshan Narasimhan, Margot Sunshine, Hong Cao, David W. Kane, Mark Reimers, Robert M. Stephens, David Bryant, Stanley K. Burt, Eldad Elnekave, Danielle M. Hari, Thomas A. Wynn, Charlotte Cunningham-Rundles, Donn M. Stewart, David Nelson, and John N. Weinstein. High-throughput gominer, an 'industrial-strength' integrative gene ontology tool for interpretation of multiple-microarray experiments, with application to studies of common variable immune deficiency (cvid). *BMC Bioinformatics*, 6:168, 2005.

[89] Zhiping Zeng, Jianyong Wang, Lizhu Zhou, and George Karypis. Coherent closed quasi-clique discovery from large dense graph databases. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–802, New York, NY, USA, 2006. ACM.

[90] Shihua Zhang, Xuemei Ning, and Xiang-Sun Zhang. Brief communication: Identification of functional modules in a ppi network by clique percolation clustering. *Computational Biology and Chemistry*, 30(6):445–451, 2006.

[91] Lizhuang Zhao and Mohammed J. Zaki. Tricluster: an effective algorithm for mining coherent clusters in 3d microarray data. In *SIGMOD '05: Proceedings of the 24th International Conference on Management of Data*, pages 694–705, New York, NY, USA, 2005. ACM.

[92] Xiaowei Zhu, Mark Gerstein, and Michael Snyder. Getting connected: analysis and principles of biological networks. *Genes and Development*, 21(9):1010–1024, 2007.

[93] Lee Zou, David Cortez, and Stephen J. Elledge. Regulation of ATR substrate selection by Rad17-dependent loading of Rad9 complexes onto chromatin. *Genes and Development*, 16(2):198–208, 2002.