MINING APPROXIMATE FREQUENT DENSE MODULES FROM MULTIPLE GENE EXPRESSION DATASETS

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

San Ha Seo

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

December 2020

Fargo, North Dakota

NORTH DAKOTA STATE UNIVERSITY

Graduate School

Title

MINING APPROXIMATE FREQUENT DENSE MODULES FROM MULTIPLE GENE EXPRESSION DATASETS

By

San Ha Seo

The supervisory committee certifies that this thesis complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Saeed Salem
Chair

Dr. Simone Ludwig

Dr. María de los Ángeles Alfonseca-Cubero

Approved:

14 April 2021
Date

Dr. Simone Ludwig
Department Chair

ABSTRACT

Large amount of gene expression data has been collected for various environmental and biological conditions. Extracting dense modules that are recurrent in multiple gene coexpression networks has been shown to be promising in functional gene annotation and biomarkers discovery. In this thesis, we propose a biclustering-based approach for mining approximate frequent dense modules. This approach reports a large number of modules with many duplicate modules. Thus, we build on this approach and propose two extended approaches for mining dense modules, which mine set of representative patterns using post-processing and on-line pattern summarization methods. The extended approaches report smaller number of modules and less duplicate modules. Experiments on real gene coexpression networks show that frequent dense modules are biologically interesting as evidenced by the large percentage of biologically enriched frequent dense modules.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my advisor, Dr. Saeed Salem, for all his support and guidance that he has given me. He has been extremely patient with me and has been a great mentor, both in academia and in life. I would like to thank the members of my thesis committee, Dr. Simone Ludwig, and Dr. Mukhlesur Rahman, for taking the time to serve on my thesis committee. I also would like to tank North Dakota State University and the Department of Compputer Science for giving me the opportunity to study and learn. Finally, I would like to thank my family for the love and support they have given me throughout writing this thesis and my life in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The analysis of complex networks plays an important role in various scientific domains, including transportation, sociology, computer science, electrical engineering, chemistry, and bioinformatics. Many real-world systems in these domains are naturally modelled as graphs. For example, an airway system is a network of airports connected via air routes. An electrical circuit may be modelled as a network of circuit elements connected by wires. The World Wide Web (WWW) is a network of web pages connected through hyperlinks. Other examples include social networks, chemical compounds, protein-protein interaction networks, and gene coexpression networks. In the field of biology and medicine, network analysis has many applications including identifying drug targets, gene function prediction, designing containment strategies for diseases, and detecting neurological disorders [22].

Adavances in DNA microarray technology made it easier and cheaper to collect and analyze tremendous amount of gene expression data. Analysis of gene expression data is useful in understanding gene function and gene regulation. Coexpressed gene are likely to be co-functional and co-regulated. Clustering genes based on coexpression has proven helpful in predicting unknown gene functions and identifying regulatory motifs [5, 2].

Gene coexpression networks, which describe the similarities in expression profiles between different genes, have attracted much attention among researchers. Graph-theoretic approaches have been applied to gene coexpression networks to analyze gene expression data. Sharan et al. [30] proposed a method to discover clusters in the gene coexpression network using graph partitioning algorithm based on minimum cut.

Due to the complex procedure of microarray experiments and the simultaneous perturbation of multiple biological pathways in the experiments, gene expression data often contains many spurious coexpression links, that is, coexpression links that have no biological significance [13]. To overcome this problem, researchers have focused on integrating multiple gene expression datasets. In this approach, each gene expression dataset is represented as a gene coexpression network and we aim to discover modules that occur frequently in the set of coexpression networks.

Several enumeration algorithms for mining frequent modules in a set of networks have been proposed [18, 33, 28]. However, pattern enumeration approaches do not scale well when applied to massive biological networks, especially when there are large frequent modules. The time and memory complexities increase exponentially with the size of modules and the number of networks [13]. To overcome this problem, many researchers have directed their attention to aggregating the networks together into a summary graph and discovering modules from it. Lee et al. [20] applied graph clustering algorithms on the summary graph to discover highly connected modules, and showed that coexpression links present in many datasets are more likely to be biologically significant.

Directly clustering a summary graph, however, may result in false positive modules, which are highly connected in the summary graph but neither frequent nor highly connected in the original networks [13]. Several algorithms have been proposed to address this problem [14, 28]. These algorithms mine frequent modules from a graph set by constructing a binary edge occurrence matrix, which is a binary matrix that describe the edge occurrences in the graph set. Then biclusters in the matrix are mined, which correspond to frequent subgraphs in the graph set.

1.1. Contribution

In this thesis, we propose a two-step algorithm to mine approximate frequent dense modules, that is, frequent dense modules that may contain noise. We mine approximate frequent modules as opposed to exact frequent modules, because mining exact frequent modules may be too strict and may fail to discover some biologically significant modules. In the first step, we construct a binary edge occurrence matrix from the graph set and then mine approximate biclusters from the matrix. In the second step, dense modules are extracted from the subgraphs induced by the biclusters.

We build on BiBit [26] algorithm to mine approximate biclusters, which are biclusters containing noise. This approach mines many edgesets with high overlap, which results in many duplicate modules. Thus we further extend the algorithm and propose two extended algorithms, which summarize the edgesets using post-processing and on-line methods respectively. The extended approaches reduce the number of edgesets and duplicate modules significantly, enabling analysis for lower support thresholds.

To demonstrate the effectiveness of our algorithm, we conducted experiments on real-world gene expression networks. Our result shows that frequent dense subgraphs are biologically interest-

ing, as large percentage of these subgraphs are biologically enriched with known KEGG pathways and molecular functions.

1.2. Thesis Overview

The remainder of the thesis is organized as follows. In Chapter 2, we present related work, including related concepts and algorithms used in our work, as well as previous algorithms proposed to mine frequent subgraphs. In Chapter 3, we present general approach, including the problem description, the general two-step framework of our algorithm, and brief description of the various techniques for mining frequent edgesets. In Chapter 4, we present the detailed description of the first approach, biclustering with noise, and the associated BiBitN algorithm, followed by the experimental results. In Chapter 5, we present the detailed description of the biclustering with post-processing summarization method, an extension of the BiBitN method, followed by the experimental results. In Chapter 6, we present the detailed description of the biclustering with on-line summarization method, another extension of the BiBitN method, followed by the experimental results. Finally, Chapter 7 presents the conclusion and future work.

2. RELATED WORK

In this chapter, we present related concepts and algorithms used in our work, including frequent itemset mining, clustering and biclustering, and graph mining. We then present previous algorithms proposed to mine frequent subgraphs in the context of biological data.

The fast growing pace of digital data collection and accumulation has necessitated the development of data mining techniques. Data mining is a process of discovering insightful and interesting patterns from large-scale data. The process generally involves collection, extraction, warehousing, and analysis of data, as well as statistics. Data mining research has evolved from, and continues to grow in various fields such as machine learning, pattern recognition, databases, statistics, artificial intelligence, business, medical, etc. Several data mining techniques have been developed, including frequent itemset mining, association rule mining, sequence mining, classification, clustering, and graph mining.

2.1. Frequent Itemset Mining

Frequent itemset mining is one of the most commonly used techniques in pattern mining. Its goal is to find sets of objects that frequently occur together. Frequent itemset mining has many applications in various areas including marketing, social media, and bioinformatics, among others. A typical application of frequent itemset mining is market basket analysis, which is mining set of items that customers frequently purchase together in supermarkets. From frequent itemsets, we can identify association rules, which are statements about the likelihood of co-occurrences of two itemsets. In the context of market basket analysis, an example of an association rule is: If

tid	Itemset
1	A B E
2	A C D E
3	B D E
4	A B C E
5	C D

(a)

sup	Itemsets
4	E
3	A, B, C, D, AE, BE
2	AB, AC, CD, CE, DE, ABE, ACE

(b)

Figure 2.1. (a) Transaction database and (b) all frequent itemsets in the database for $minsup = 2$.

a customer purchased items a and b, she is likely to purchase item c. Frequent itemsets and association rules can then be utilized to help plan marketing strategies.

Here we briefly explain frequent itemset mining. Let $\mathcal{I} = \{x_1, x_2, \ldots, x_m\}$ be a set of all items. Any subset of $\mathcal{I}$ is called an itemset. A transaction is a tuple of the form $\langle t, X \rangle$, where t is a unique transaction id or *tid*, and X is an itemset. A set of transactions is called a transaction database. Figure 2.1 (a) shows an example of a transaction database containing five transactions. Note that we drop the set notation for convenience.

Given a transaction database D, the support of an itemset X, denoted $sup(X)$, is the number of transactions in D that contains X. An itemset X is frequent if $sup(X) \geq minsup$, where $minsup$ is a user-defined minimum support threshold. Figure 2.1 (b) shows all frequent itemsets, grouped by support values, in the given transaction database for $minsup = 2$. Given a minimum support threshold $minsup$, the task of frequent itemset mining is to find all frequent itemsets in the database.

The brute-force approach of enumerating all possible itemsets to find frequent itemsets has exponential time complexity, which is too expensive for large databases. In order to overcome this problem, we can employ the anti-monotone property of the support of an itemset, which states that the support of an itemset is monotone decreasing as the itemset gets expanded to a larger itemset [9]. In other words, all subsets of a frequent itemset must be frequent, and all supersets of an infrequent itemset must be infrequent. This property allows to prune all supersets of infrequent itemsets and significantly reduce the search space.

For a large database or a low minimum support threshold, the number of frequent itemsets may be too large to report, and it is necessary to report a condensed representation of the frequent itemsets. One such representation is called the maximal frequent itemsets. A maximal frequent itemset is a frequent itemset that does not have a frequent superset. Due to the anti-monotone property of the support of an itemset, every subset of a maximal frequent itemset is frequent. Thus, from the set of all maximal frequent itemsets, $\mathcal{M}$, we can recover the set of all frequent itemsets. However, we are unable to determine the support of each frequent itemset from $\mathcal{M}$. If we are only interested in mining the set of all frequent itemsets without their support values, reporting $\mathcal{M}$ would suffice. In the exmaple given in Figure 2.1, the maximal frequent itemsets are CD, DE, ABE, and ACE.

Algorithm 1 GenMax Algorithm

Initial Call:
$\mathcal{M} = \emptyset$
$P = \{\langle i, t(i)\rangle | i \in \mathcal{I}, sup(i) \geq minsup\}$

GenMax$(P, minsup, \mathcal{M})$:
1. $Y = \cup X_i$
2. **if** $\exists Z \in \mathcal{M}$, such that $Y \subseteq Z$ **then**
3. **return**
4. **end if**
5. **for** each $\langle X_i, t(X_i)\rangle \in P$ **do**
6. $P_i = \emptyset$
7. **for** each $\langle X_i, t(X_i)\rangle \in P, with j > i$ **do**
8. $X_{ij} = X_i \cup X_j$
9. $t(X_{ij}) = t(X_i) \cap t(X_j)$
10. **if** $sup(X_{ij}) \geq minsup$ **then**
11. $P_i = P_i \cup \{\langle X_{ij}, t(X_{ij})\rangle\}$
12. **end if**
13. **end for**
14. **if** $P_i \neq \emptyset$ **then**
15. $GenMax(P_i, minsup, \mathcal{M})$
16. **else if** $\nexists Z \in \mathcal{M}, X_i \subseteq Z$ **then**
17. $\mathcal{M} = \mathcal{M} \cup X_i$
18. **end if**
19. **end for**

2.1.1. GenMax Algorithm

GenMax [11] is a maximal frequent itemset mining algorithm used in this work. Here we briefly explain the GenMax algorithm. We define an itemset-tidset pair, or IT-pair, as a tuple of the form $\langle X, t(X)\rangle$ where X is an itemset and $t(X)$ is the tidset of transactions containing X. The support of an itemset X is the cardinality of its tidset, that is, $sup(X) = |t(X)|$. The algorithm keeps track of tidsets of candidate itemsets to help compute support values. For example, in the transaction database in Figure 2.1 (a), $t(AE) = \{1, 2, 4\}$ (124 for short), and $sup(AE) = |t(AE)| = 3$. In general, the tidset of the union of two itemsets is equal to the intersection of the tidsets of the individual itemsets, that is, given two itemsets X_a and X_b, $t(X_{ab}) = t(X_a) \cap t(X_b)$, where $X_{ab} = X_a \cup X_b$. This property is used to quickly compute tidset of a candidate itemset from the tidsets of its subsets.

The GenMax algorithm is illustrated in Algorithm 1. The recursive GenMax method takes a set of frequent itemsets (with tidsets), P, and the set of discovered maximal itemsets, $\mathcal{M}$. As we will see, all itemsets in P share a common prefix. The algorithm first checks if the union of all itemsets in P has a superset in $\mathcal{M}$. If so, the current branch is pruned because no maximal itemset

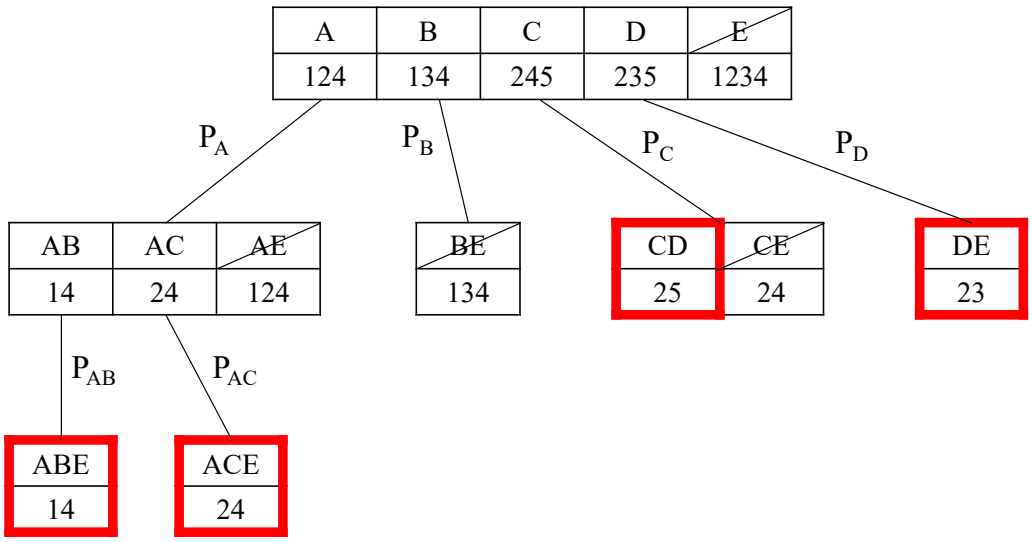


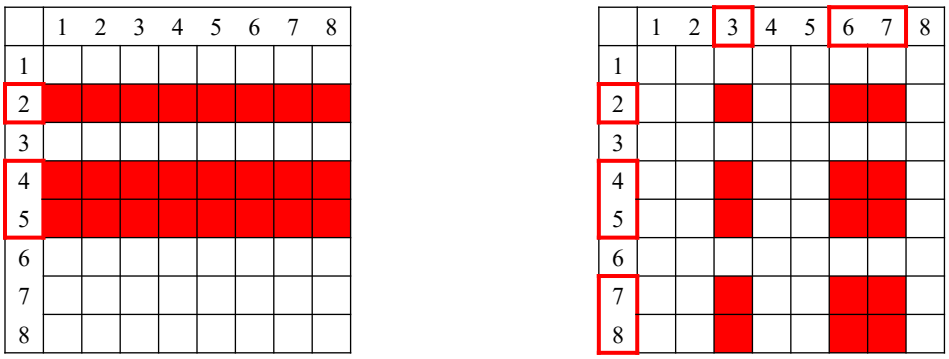
Figure 2.2. Execution of GenMax algorithm on the database in Figure 2.1 (a) for $minsup = 2$. Maximal itemsets are shown with red outlines, and pruned branches are shown with diagonal lines.

can be generated from the current branch. If not, the algorithm extends each itemset X_i in P by taking the union with all other itemsets X_j in P to generate new candidates X_{ij}. The tidset $t(X_{ij})$ is computed by intersecting $t(X_i)$ and $t(X_j)$, and we compute the support from the tidset. That is, $sup(X_{ij}) = |t(X_{ij})| = |t(X_i) \cap t(X_j)|$. The candidates X_{ij} are added to the set P_i, if frequent. If P_i is not empty, GenMax is called recursively with P_i to find potential extensions of X_i. If P_i is empty, X_i is potentially maximal, so it is added to $\mathcal{M}$ if it has no superset in $\mathcal{M}$. GenMax is initially called with the set of all frequent items (with tidsets) as P and empty set $\mathcal{M}$. After the run, the set $\mathcal{M}$ will contain all maximal frequent itemsets.

Figure 2.2 illustrates the execution of GenMax alogrithm on the database in Figure 2.1 (a) for $minsup = 2$. The root of the tree represents the initial call with the set of all frequent single items and their tidsets. The maximal frequent itemsets are indicated by the red outlines and the pruned candidates are indicated by the diagonal lines. For example, the candidate AE is pruned because it cannot be extended and its superset ACE is already in $\mathcal{M}$. Infrequent itemsets are not shown.

2.2. Clustering

In order to learn from data, we define a set of features that describe data objects and compare the data objects based on similarities/dissimilarities among them with respect to these features. In clustering, also called unsupervised learning, we try to separate the dataset into groups



(a) Clustering (b)Biclustering

Figure 2.3. Clustering and biclustering in matrix representation. (a) shows a cluster $\{2, 4, 5\}$. (b) shows a bicluster $\{2, 4, 5, 7, 8\}$ (for columns $\{3, 6, 7\}$) which is a local pattern not found in (a).

such that elements have high intra-group similarities and low inter-group similarities. In clustering, the data objects are not labeled with predefined classes, as is the case in supervised learning. The goal is to find 'natural' hidden clusters in the dataset. Here we explain several commonly used clustering approaches.

K-means [21] clustering algorithm starts out by partitioning the dataset into k random nonempty clusters and computing the centroid of each cluster. Then each data object is assigned to the nearest cluster and new centroids are computed. The process repeats until it converges. K-means algorithm is efficient but has many limitations. For instance, the algorithm requires the prior specification of the number of clusters, and is unable to discover overlapping clusters.

Hierarchical clustering algorithms produce a set of nested clusters organized as a hierarchical tree. The reults of hierarchical clustering can be visualized as a dendrogram, and it can be cut at different levels to obtain different number of clusters. Two main types of hierarchical clustering algorithms are agglomerative and divisive. Agglomerative clustering starts with each data object as an individual cluster, and repeatedly merges the closest pair of clusters until only one cluster is left. Divisive clustering starts with the entire dataset as a single cluster, and repeatedly split clusters until each cluster contains one data object.

Other approaches cluster datasets based on density. As an example, the DBSCAN [6] algorithm builds a cluster by including data objects that have dense neighborhood in the data space (core points), and all objects in their neighborhood (border points). The density-based

clustering algorithms are highly scalable, able to discover clusters of arbitrary shapes, and able to handle noisy data.

A dataset are typically represented by a matrix in which each row represents a data object and each column represents an attribute(feature) that the data objects may have. Data matrices can be real valued or binary valued. Real matrices are useful for representing experimental measurements, as they are usually real valued. For example, a gene expression dataset is usually represented as a real matrix where each entry is the gene expression level of the corresponding gene in the corresponding sample. Binary matrices are useful when representing relationships between data objects and their possible attributes. For example, a binary matrix can be used to simplify a gene expression dataset to represent relationships between genes and samples based on expression/non-expression [25]. In another example, a binary matrix is used to represent relationships between genes and proteins based on coding/non-coding [19].

In matrix represetation, the similarity between two data objects correspond to the similarity between the two corresponding rows. Different row similarity measures may be employed in different contexts. High row similarity means the rows have similar profiles. In matrix representatoin, clustering a dataset means grouping highly similar rows together. Figure 2.3 (a) shows an example of clustering in matrix representation. Rows 2, 4, and 5 have high similarities, thereby forming a cluster $\{2, 4, 5\}$. It is also possible to cluster based on columns, which would correspond to clustering based on attributes across all data objects.

2.3. Biclustering

The effectiveness of the standard clustering methods is limited because, in general, a group of data objects may exhibit similar patterns under only a subset of attributes. The same limitation exists when clustering based on attributes. In gene expression data, for instance, a subset of genes is generally coexpressed under only a subset of samples. This limitation has led to the development of biclustering algorithms. A biclustering algorithm clusters rows and columns simultaneously, and thus is able to extract local patterns that are not discovered by the standard clustering methods.

In matrix representation, biclustering a dataset means finding submatrices with high row (or column) similarities, as illustrated in Figure 3b. Figure 2.3 (b) shows an example of biclustering in matrix representation. Rows 2, 4, 5, 7, and 8 have high similarities in columns 3, 6, and 7, thereby forming a bicluster $\{\{2, 4, 5, 7, 8\}, \{3, 6, 7\}\}$, which is a local pattern not found in (a).

Algorithm 2 BiBit Algorithm

Input:
$\mathcal{B}$: $m \times n$ binary matrix; m edges $\times$ n graphs
mnr: minimum number of rows
mnc: minimum number of columns
Output:
$\mathcal{X}$: set of final biclusters

```
1.    for every edge pair (i, j) do
2.      S(i, j) = S(i) ∩ S(j)
3.      if S(i, j) is new and |S(i, j)| ≥ mnc then
4.        ⟨I, S(i, j)⟩ = ⟨{i, j}, S(i, j)⟩
5.        for every remaining edge, q ∈ E \ I   do
6.          if   S(q) ∩ S(i, j) = S(i, j) then
7.              I = I ∪ q
8.          end if
9.        end for
10.       if |I| ≥ mnr then
11.         X = X ∪ ⟨I, S(i, j)⟩
12.       end if
13.     end if
14.   end for
```

Different approaches are employed for extracting biclusters from real datasets and binary datasets. Some biclustering algorithms for real datasets are described in [17, 31]. We will not describe these algorithms here, since we are primarily interested in biclustering binary data in this work.

Several biclustering algorithms for binary datasets have been developed [19, 32, 26]. In [19] and [32], biclusters are defined as submatrices with high density of ones. The common theme of these algorithms is to define an objective function such that a bicluster with a high score tends to be large in size and have high density of ones. In an iterative process, a bicluster that maximizes the objective function is found. These algorithms can only find one bicluster in each run. Therefore, the discovered bicluster is masked from the input matrix in the subsequent runs. As the result, these algorithms are unable to discover overlapping biclusters.

BiBit [26] is another biclustering algorithm for binary data. BiBit algorithm is capable of discovering overlapping biclusters and is easily extended to handle noisy data. Modified versions of BiBit (BiBitN and BiBitNS) are used in this work.

2.3.1. BiBit Algorithm

BiBit (Bit-Pattern Biclustering) [26] is a biclustering algorithm for binary datasets. It is relatively efficient and robust to density and size of input data. The algorithm is illustrated in

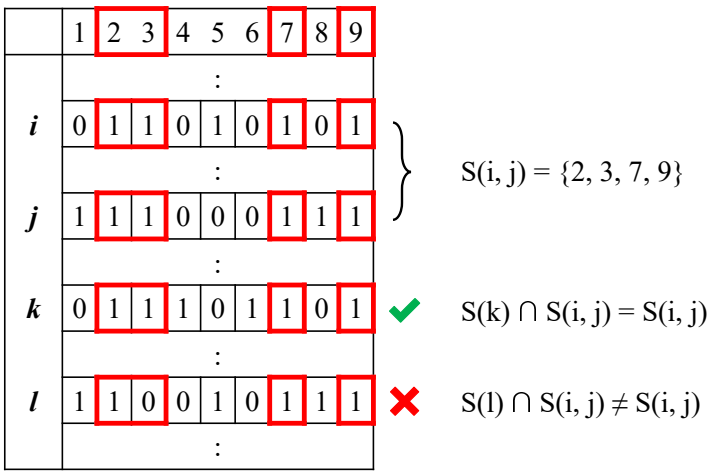
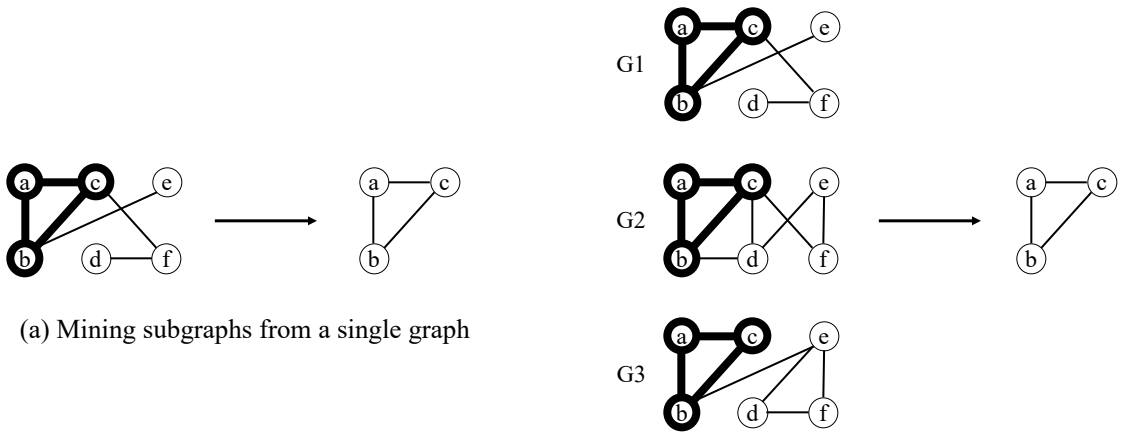


Figure 2.4. Pattern extension in BiBit. Rows i and j produce seed $S(i,j) = \{2,3,7,9\}$. Row k is added to the pattern because $S(k) \cap S(i,j) = S(i,j)$, while row l is not added to the pattern because $S(l) \cap S(i,j) \neq S(i,j)$.

Algorithm 2. Given an $m \times n$ binary matrix $\mathcal{B}$ in which rows correspond to edges and columns correspond to graphs, let $S(i)$ denote the set of column (graph) indices j such that $\mathcal{B}_{ij} = 1$, i.e., $S(i) = \{j \mid \mathcal{B}_{ij} = 1\}$, and let $S(i,j) = S(i) \cap S(j)$. For example in Figure 2.4, $S(i) = \{2,3,5,7,9\}$, $S(j) = \{1,2,3,7,9\}$, and $S(i,j) = \{2,3,7,9\}$. BiBit algorithm selects a pair of rows i and j and generates the bit-pattern $\langle\{i,j\}, S(i,j)\rangle$. A bit-pattern is a tuple of the set of rows and its supporting columns. The bit-pattern $\langle\{i,j\}, S(i,j)\rangle$ is used as a seed for a bicluster and $S(i,j)$ represents the column set for the bicluster. Each remaining row q is added to the bicluster if $S(q) \cap S(i,j) = S(i,j)$, that is, if the set of columns in which object q appears is a superset of the set of columns in which both objects i and j appear. In Figure 2.4, row k is added to the bicluster from seed $S(i,j)$ because $S(k) \cap S(i,j) = S(i,j)$, and row l is not added to the bicluster because $S(l) \cap S(i,j) \neq S(i,j)$. The result is a bicluster that consists of all ones, and its columns is the seed $S(i,j)$. This process is repeated for every pair of rows as a seed. Only bicluster with at least mnc columns and mnr rows are returned.

While the BiBit algorithm is efficient, it has several limitations. One problem is that it may not discover every bicluster that satisfies the given condition. This happens because when the size of a seed is much larger than mnc, it will miss rows that satisfy the mnc requirement but do not match the seed. For example, if the seed appear in 90% of the graphs, this seed can only be extended with edges that appear in the same set of graphs, and thus limiting the number of



(a) Mining subgraphs from a single graph

(b) Mining subgraphs from multiple graphs

Figure 2.5. Two types of graph mining problems. (a) Mining interesting subgraphs from a single large graph. (b) Mining frequently occurring interesting subgraphs from a set of multiple graphs.

possible extensions available for the seed. This is a minor problem if the number of columns in the input binary matrix is small, but it becomes more apparent if the number of columns is large. Another limitation of the BiBit algorithm is that it is only able to discover exact biclusters, which are biclusters consisting of all ones. The BiBit algorithm may be extended to discover approximate biclusters, that is, biclusters that may contain some noise, by relaxing the condition for extending the bicluster.

2.4. Graph Mining

In traditional data analyses, data objects are assumed to be independent of one another. However, this is not the case in reality, since data objects are often connected to one another by various relationships to form a network. Networks are ubiquitous in today's world. The internet, the World Wide Web, and social networks are some examples of networks. Interactions among proteins in biological processes can be modelled as a network of proteins [18]. A gene expression data can be represented as a network of coexpressed genes [33]. Network data can be modelled as a graph. Graph is a mathematical structure consisting of a set of nodes, and edges that connect the nodes. In graph representation of a network, each node corresponds to a data object and each edge corresponds to a connection between two objects.

Due to the increasing availability of network data, graph mining has become an important reserch area in data mining. The goal of graph mining is to discover interesting subgraphs from a single or multiple graphs. We are usually interested in finding highly connected subgraphs (dense

Algorithm 3 DME Algorithm

Input:
V: node set
M: adjacency matrix
θ: minimum density
U: current module (DME is called with $U = \emptyset$)
Output:
locally maximal modules that satisfy the density threshold

```
1.   DME (V, M, θ, U):
2.     locallyMaximal = true
3.     for each v ∈ VnU do
4.       if ρ(U ∪ {v}) ≥ θ then
5.         locallyMaximal = false
6.         if U ∪ {v} is child of U then
7.           DME (V, M, θ, U ∪ {v})
8.         end if
9.       end if
10.    end for
11.    if locallyMaximal then
12.      output U
13.    end if
```

modules, cliques, etc.). The highly connected subgraphs may represent different concepts in different contexts. For example in a social network, a highly connected subgraph may represent a social community. In a gene coexpression network, one may represent a group of genes that are co-functional and/or co-regulated [5, 2].

There are two main types of graph mining problems. One type is mining subgraphs from a single large graph, as shown in Figure 2.5 (a). Several approaches have been proposed to tackle this type of problems [3, 23, 1, 7, 8]. [3] and [23] recursively partition the graph based on betweenness and modularity, respectively, into a set of modules. [1] and [7] define module criteria and use heuristic search techniques to find modules. The DME [8] algorithm uses an enumeration technique to find the complete set of modules that satisfy a user-defined density threshold.

Another type of graph mining problems is mining subgraphs from a set of multiple graphs, as shown in Figure 2.5 (b). The goal here is to discover subgraphs that occur frequently in the set of graphs. Methods to tackle this type of problems are described in Section 2.5.3 in the context of biological data.

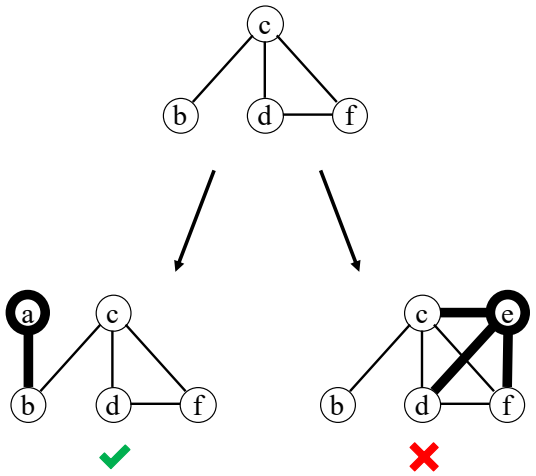


Figure 2.6. Parent-child relationship in DME algorithm. The left subgraph is a child of the graph above, because the degree of the new node a is less than or equal to every other node already in the graph. The right subgraph is not a child of the graph above, because the degree of the new node e is greater than some nodes already in the graph.

2.4.1. DME Algorithm

The DME (Dense Module Enumeration) [8] algorithm mines dense subgraphs in a single large graph. Given a user-defined density threshold, DME returns all locally maximal subgraphs that satisfy the threshold. The original DME algorithm is applicable to weighted graphs and is capable of integrating constraints from additional data sources. In this work, we apply DME on unweighted graphs without additional constraints. Thus, we describe a simplified version of DME, which applies to unweighted graphs and does not consider additional constraints.

DME is an enumeration algorithm, which starts with the empty set and iteratively grows the set by adding one element at a time. Conventional pruning strategies do not work because, in general, a superset of a module may have a higher density than the module itself. The key idea of DME is to use the parent-child relationship for modules, which makes pruning possible.

The parent-child relationship for modules is defined as follows: A unique parent of a module is obtained by removing the node with the minimum degree. A strict total ordering (e.g. lexicographic order) on nodes is defined, in order to break ties when there are multiple nodes with the minimum degree. Graph has a convenient property that adding a node with degree less than or equal to the degree of every other node already in the module will not increase the density of the module. In terms of parent-child relationship, this means the density of a child module is always less than or equal to its parent's. The parent-child relationship in DME algorithm is illustrated in

Figure 2.6. The left subgraph is a child of the graph above, because the degree of the new node a is less than or equal to every other node already in the graph. The right subgraph is not a child of the graph above, because the degree of the new node e is greater than some nodes already in the graph. The density of the parent graph is 0.67, while the densities of the left and right subgraphs are 0.5 and 0.7 respectively, therefore the property holds.

The DME algorithm is illustrated in Algorithm 3. In each iterations, it only generates the children of each module, instead of all possible extensions of the module. This way, the module density decreases monotonically along each path from root to leaf in the enumeration tree. The children that violate the density threshold can be pruned. The discovered modules are locally maximal, but in practice they are likely to be globally maximal as well.

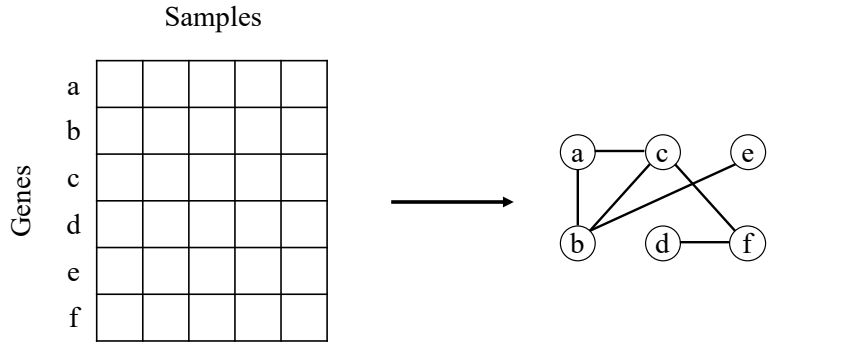
2.5. Clustering Gene Expression Data

Adavances in DNA microarray technology have enabled the collection and analysis of huge amount of gene expression data. Gene expression datasets can be clustered by genes, samples, or both genes and samples simultaneously. In gene-based clustering, each cluster of genes may correspond to co-functional and/or co-regulated genes. In sample-based clustering, each cluster may correspond to disease or cancer types. In general, only a subset of genes and a subset of samples are related to a particular biological process [16]. Therefore, in biclustering, genes and samples are clustered simultaneously. In this work, we focus on gene-based clustering.

Clustering coexpressed genes have proven useful in understanding gene function and gene regulation. Coexpressed genes are likely to have similar biological functions, and clustering coexpressed genes can help predict previously unknown gene functions based on the genes with known functions in the same cluster [5]. Coexpressed genes are also likely to be co-regulated. Clustering coexpressed genes can help identify regulatory motifs by searching for common DNA sequences at the promoter regions of the genes in the same cluster [2].

2.5.1. Gene Coexpression Networks

Gene expression data from DNA microarray experiments are typically presented as real matrix whose rows correspond to genes and columns to samples. The value of the matrix entry M_{ij} represents the expression level of gene i in sample j, and row i represents the expression profile of gene i. Figure 2.7 (a) shows the structure of a gene expression dataset with six genes and five samples.



(a) Gene expression dataset (b) Gene coexpression network

Figure 2.7. (a) Gene expression dataset of six genes and five samples. (b) Gene coexpression network constructed from a dataset of six genes.

Gene expression datasets are often represented as gene coexpression networks for analysis. A gene coexpression network is a graph in which each node corresponds to a gene and there is an edge between two nodes if the two genes have similar expression profiles in the dataset. Typcially, Pearson correlation coefficient is used as the similarity measure for expression profiles [34]. If the similarity value between two genes is greater than a user-defined similarity threshold, then we add an edge between the two genes in the coexpression network. Figure 2.7 (b) shows an example of a gene coexpression network constructed from a dataset of six genes. The presence of the edge (a, b), for example, implies that the genes a and b have similar expression profiles in the dataset. A gene coexpression network can be weighted. In the case of weighted network, each edge weight is the similarity value between the two genes that the edge connects.

After constructing gene coexpression networks, graph-theoretic methods can be used to analyze the data. For example, a highly connected region in the coexpression network may correspond to a module of coexpressed genes.

2.5.2. Mining Single Gene Expression Dataset

Various conventional clustering methods have been used to identify gene groups in gene expression data, including k-means and hierarchical approaches [12, 5]. While these general-purpose algorithms have proven useful in some applications, they have not been effective in many applications [16]. Several algorithms designed specifically for gene expression data have been proposed [15, 30]. The DHC [15] algorithm proposed a density-based approach to identify clusters from gene expression data. In [30], a weighted gene coexpression network was constructed from the gene

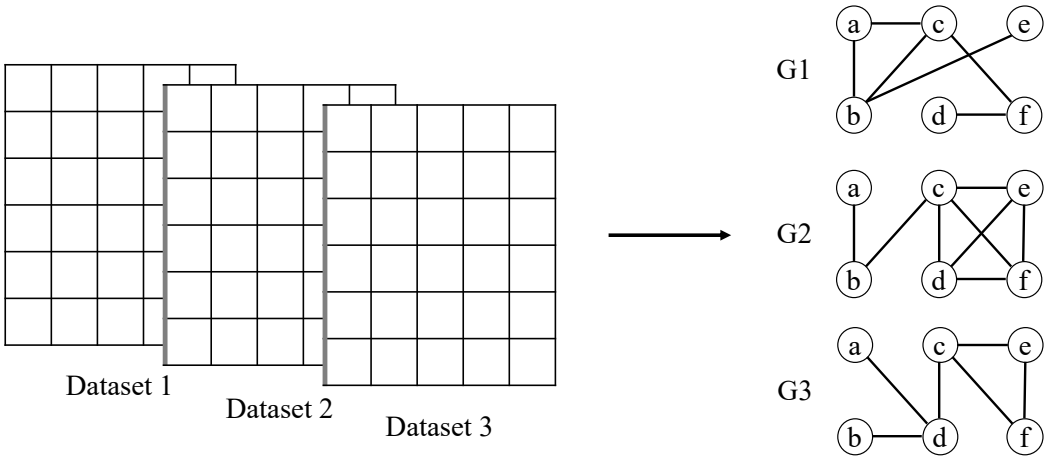


Figure 2.8. Integrating multiple gene expression datasets. Multiple datasets are represented as a set of gene coexpression networks, where each dataset corresponds to one coexpression network.

expression data and was split recursively based on minimum cut to find clusters. It has been shown that these algorithms perform better than the conventional approaches on some datasets.

2.5.3. Integrating Multiple Gene Expression Datasets

For experimental reasons, many coexpression links are of questionable biological relevance. Due to the complex procedure of microarray experiments, gene expression data often contains a lot of noise, leading to a significant number of spurious coexpression links [13]. Additionally, some coexpression may be caused by the simultaneous perturbation of multiple biological pathways in the particular experiment rather than by biological relevance [14]. These spurious coexpression links may lead to the discovery of false modules.

To overcome the problem of spurious links, recent studies have focused on integrating multiple gene expression datasets to discover gene clusters that appear across multiple datasets. It is based on the expectation that biological modules are active across multiple datasets. Many of these studies used graph-theoretic approach, where multiple gene expression datasets are represented as a set of gene coexpression networks. Each dataset corresponds to one coexpression network, as shown in Figure 2.8. The gene clustering task is then equivalent to mining frequently occurring modules in the set of multiple gene coexpression networks. Biological networks often have the convenient property where each node has a unique label. This property can be exploited to avoid the subgraph isomorphism problem and greatly simplify the task.

Several enumeration algorithms for mining frequent modules in a set of graphs have been proposed. MULE [18] is a pattern enumeration algorithm based on a depth-first frequent itemset mining approach. We are usually interested in mining modules that are not only frequent but also highly connected, because data objects (genes, proteins, etc.) in a highly connected module may have higher biological relevance than those that are not. Therefore, some algorithms have imposed additional constraints to mine frequent highly-connected modules. In [33], a connectivity constraint was imposed to mine frequent closed subgraphs. In [15], a density constraint was imposed to mine frequent quasi-cliques.

Pattern enumeration approaches do not scale well when applied to massive biological networks, especially when there are large frequent modules. The time and memory complexities of these algorithms increase exponentially with the size of modules and the number of networks [13]. To address the scalability issue, many studies have focused on aggregating the networks together and discovering modules in the aggregated graph. In [20], the authors combined frequent coexpression links in multiple coexpression networks to build a summary graph, and applied hierarchical clustering and the MCODE [1] algorithms on the summary graph to discover highly connected modules. It was shown that coexpression links present in many datasets are more likely to represent known functional relevance. However, directly clustering an aggregated graph may result in the discovery of false positive modules, which are not dense in the original set of networks. The edges in these modules may be scattered across the networks such that they are highly connected in the aggregated graph, but neither frequent nor highly connected in the original networks [13]. Several algorithms have been proposed to address this problem [13, 14, 28].

The CODENSE [13] algorithm efficiently mines coherent dense subgraphs across large number of graphs. A coherent subgraph is a subgraph whose edges show highly correlated occurrences across the whole graph set. The algorithm first builds a summary graph and mines the dense subgraphs in the summary graph. Then for each dense summary subgraph, it constructs the second-order graph, which illustrates the co-occurrence of edges across the original graph set. Finally, dense subgraphs in the second-order graph are extracted as the final results. The algorithm is able to overcome the false positive module problem by exploiting the property that a coherent subgraph's second-order graph must be dense. Another notable feature of CODENSE is its ability

to mine overlapping modules. It is important because, in general, one gene may be involved in multiple biological processes.

In the approach proposed by Huang et al. [14], the coexpression graphs are represented as a binary edge occurrence matrix, which is a binary matrix where each row corresponds to an edge, each column corresponds to a graph, and each entry indicates the presence of the edge in the graph. Frequent itemset mining technique is employed to mine frequent edgesets from the edge occurrence matrix. These frequent edgesets serve as seeds for a biclustering algorithm, which uses simulated annealing to maximize an objective function such that the discovered biclusters are large and have high density of ones. Connected components in the biclusters are returned as the final output modules. The discovered modules are frequent but not necessarily highly connected.

The MFMS [28] algorithm mines maximal frequent collections of k-cliques and percolated k-cliques across many coexpression networks. The coexpression networks are first represented as the binary edge occurrence matrix. Maximal itemset mining algorithm (GenMax [11]) is then used on the edge occurrence matrix to mine maximal frequent edgesets. Cliques and percolated cliques are extracted from the subgraph induced by each maximal frequent edgeset. Mining maximal frequent edgesets is equivalent to mining exact biclusters, that is, biclusters whose entries are all ones. This means the discovered modules cannot contain noise. This condition may be too strict and may fail to identify biologically relevant modules that contain some noise due to the noisy nature of gene expression data. Salem et al. [29, 27] proposed an approach that constructs a weighted network whose nodes corresponds to the original edges in the coexpression networks. The weight between two edges is calculated as a combined score based on the topological similarity between the edges and the occurrence similarity.

3. GENERAL APPROACH

In this chapter, we present the problem description and the general two-step framework of our algorithm. We also present brief description of the various techniques for mining frequent edgesets, including frequent itemset mining, biclustering (BiBit), biclustering with noise (BiBitN), biclustering with post-processing summarization, and biclustering with on-line summarization. BiBitN and the two extended algorithms (post-processing and on-line summarization) are described in more detail in chapters 4, 5, and 6.

3.1. Problem Description

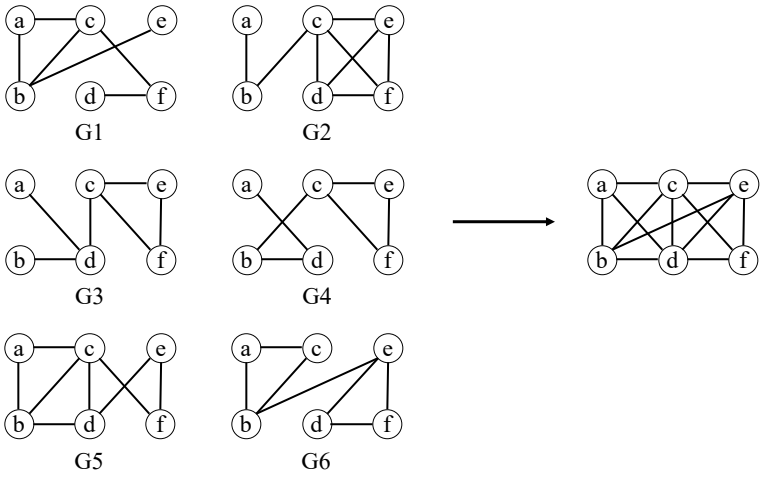
In this section, we introduce preliminary definitions that we use throughout the paper, and describe the problem of mining approximate frequent dense modules from a relation graph set.

Gene coexpression networks have a property that each gene occurs only once in the network. This type of network can be modelled as a relation graph, where each node has a unique label. A relation graph set is a set of graphs that share a common set of nodes.

Relation Graph Set: A relation graph set is a set of n graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ where $G_i = (V, E_i)$ and $E_i \subseteq V \times V$. A common set of nodes V is shared by all graphs.

Figure 3.1 (a) shows an example of a relation graph set of six graphs. Note they share a common set of nodes. To set a common framework for the discussion of the different methods, we represent the n graphs as a summary graph $G(V, E)$ and an associated binary edge occurrence matrix, $\mathcal{B}$. Each row of the matrix is a binary vector whose entries represent the presence/absence of the edge in the corresponding graph.

Summary Graph and Edge Occurrence Matrix: Given a relation graph set $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ where $G_i = (V, E_i)$, let $E = \{e_1, e_2, \ldots, e_m\} = \bigcup_{i=1}^{n} E_i$. The edge occurrence matrix $\mathcal{B}$ is an $m \times n$ binary matrix where $\mathcal{B}_{ij} = 1$ if $e_i \in E_j$; 0 otherwise. The relation graph set can be represented as $\mathcal{G} = (V, E, \mathcal{B})$.



	G1	G2	G3	G4	G5	G6
a-b	1	1	0	0	1	1
a-c	1	0	0	0	1	1
a-d	0	0	1	1	0	0
b-c	1	1	0	1	1	1
b-d	0	0	1	1	1	0
b-e	1	0	0	0	0	1
c-d	0	1	1	0	1	0
c-e	0	1	1	1	0	0
c-f	1	1	1	1	1	0
d-e	0	1	0	0	1	1
d-f	1	1	0	0	0	1
e-f	0	1	1	1	1	1

(a) Relation Graph Set (b) Summary Graph and Edge Occurrence Matrix

Figure 3.1. (a) Relation graph set of six graphs with six nodes. (b) Summary graph and the binary edge occurrence matrix for the relation graph set in (a).

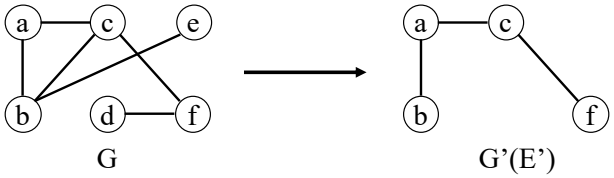


Figure 3.2. $G'(E')$ is the subgraph of G induced by the edgeset $E' = \{(a, b), (a, c), (c, f)\}$.

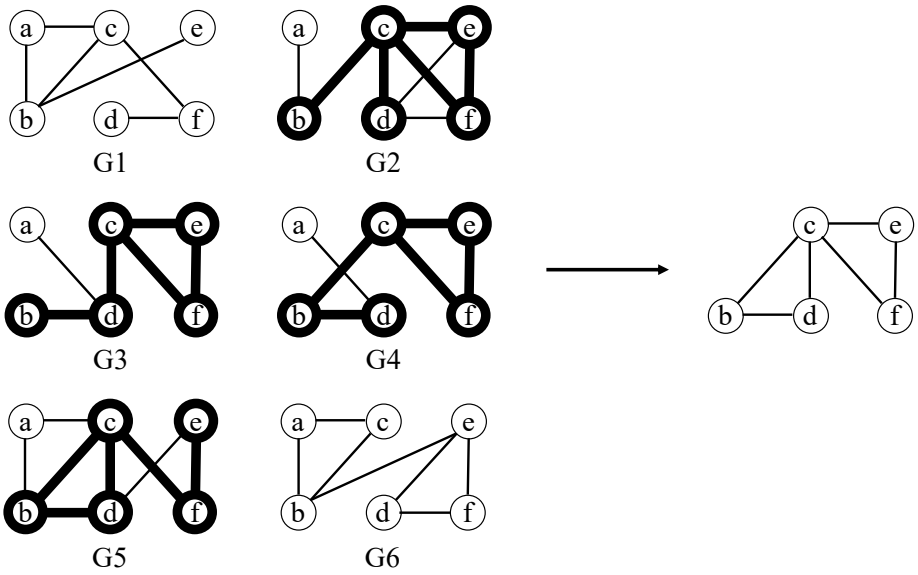


Figure 3.3. Approximate frequent subgraph of the relation graph set in Figure 3.1 (a) for $minsup = 4$ and $r = 0.25$.

Figure 3.1 (b) illustrates the summary graph and the associated binary edge occurrence matrix for the relation graph set in (a). For example, the first row of the edge occurrence matrix shows that the edge (a, b) is present in graphs $\{G1, G2, G5, G6\}$.

Edge-Induced Subgraph: Given a graph $G(V, E)$ and an edgeset $E' \subseteq E$, the edge-induced subgraph $G'(V', E')$ of G (induced by edgeset E') is a graph whose edgeset is E' and the node set is $V' = \bigcup V(e)$ for all $e \in E'$ where $V(e)$ denotes the endpoints of e.

Figure 3.2 shows an example of an edge-induced subgraph. $G'(E')$ is the subgraph of G induced by the edgeset $E' = \{(a, b), (a, c), (c, f)\}$.

Note that an edge-induced subgraph does not have isolated nodes since each node that is present in the induced subgraph is an endpoint of at least one edge. Since an edge-induced subgraph is uniquely identified by its edgeset, we refer to the frequent edge-induced subgraph as a frequent edgeset.

Frequent Subgraph: Given a graph set $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$, a minimum support threshold *minsup*, an edge-induced subgraph G' is a frequent subgraph if it is a subgraph of at least *minsup* graphs. A subgraph $G'(V', E')$ is a subgraph of $G = (V, E)$, denoted as $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E$. The supporting graphs of a subgraph is the set of graphs in which the subgraph appears.

$$sup(G', \mathcal{G}) = \{G_{i1}, G_{i2}, \cdots, G_{ik}\}$$

such that G' is a subgraph of G_i for each G_i in $sup(G', \mathcal{G})$ and k is the number of graphs in which the subgraph appears. When the graph dataset is understood from the context, we denote $sup(G', \mathcal{G})$ simply as $sup(G')$. A subgraph G' is frequent in a graph set $\mathcal{G}$ if the number of supporting graphs is at least *minsup* graphs, i.e., $|sup(G', \mathcal{G})| >= minsup$.

The definition of frequent subgraphs requires all the edges of a subgraph to appear in all the supporting graphs. Given that some of edges might be missing from a coexpression network due to noise and correlation cutoff, we should change the definition of the occurrence of a subgraph

in a coexpression network. Thus we relax the occurrence constraint and introduce the approximate frequent subgraph that is a relaxed form of the frequent subgraph by allowing missing edges (noise). An edge is approximately supported by a graph set if the edge appears in most of the graphs, and a subgraph is approximately supported by a graph set if all the edges of the subgraph are approximately supported by the same graph set.

Approximate Frequent Subgraph: Given a relation graph set $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$, a minimum support threshold $minsup$, and a noise ratio r, an edge-induced subgraph $G'[E']$ is an approximate frequent subgraph if and only if there exists a graph set $D \subseteq \mathcal{G}$ such that $|D| >= minsup$ and for every edge $e \in E'$, e occurs in at least $[\,]|D| * (1 - r)$ graphs in D, the nearest integer to $|D| * (1 - r)$.

The noise ratio r is a real number between 0 and 1, and represents the ratio of the allowed noise. A higher value of r means more noise is allowed. In our definition of approximate frequent subgraph, an edge e need not be present in every graph in D, as long as it occurs in most of them. For example, the graph in Figure 3.3 is an approximate frequent subgraph of the relation graph set in 3.1 (a) for $minsup = 4$ and $r = 0.25$, because every edge in the graph occurs in at least three out of the four graphs in $\{G2, G3, G4, G5\}$.

In this work, we are interested in discovering frequent subgraphs that are dense. Both the density and the recurrence of the subgraph increase the likelihood that the subgraph is biologically meaningful.

Graph Density: The density of a graph G is $2m/(n(n-1))$ where m is the number of edges and n is the number of nodes in G. G is dense if its density is greater than or equal to a minimum density threshold.

Our problem is formulated as follows: Given a relation graph set $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$, minimum support, noise, and density thresholds, find subgraphs that are both approximate frequent and dense.

We follow a two-step approach to mine frequent dense subgraphs. The first step is to mine frequent subgraphs from the relation graph set, and the second step is to mine dense modules from the subgraphs.

3.2. Mining Frequent Edgesets

The first step in our two-step approach is to mine frequent subgraphs from the relation graph set. The general approach we take to mine frequent subgraphs is to first represent the relation graph set as the summary graph and the associated binary edge occurrence matrix. The problem is then equivalent to mining frequent edgesets (row sets) from the edge occurrence matrix, since the subgraphs induced by the frequent edgesets are the frequent subgraphs of the relation graph set.

In this section, we introduce five approaches to mine frequent edgesets: frequent itemset mining, biclustering (BiBit), biclustering with noise (BiBitN), biclustering with post-processing summarization, and biclustering with on-line summarization. In the later chapters, we describe in detail the BiBitN, post-processing summarization, and on-line summarization methods, which are the main methods we propose in this work.

3.2.1. Frequent Itemset Mining

Here, we briefly describe how frequent edgesets are mined using frequent itemset mining. Let $E = \{e_1, e_2, \ldots, e_{|E|}\}$ be the set of the union of all the edges in the graphs. The graph set is represented as a set of transactions defined over the set of edges in the graphs. Each graph is essentially a subset of edges from the entire edgesets, i.e., $E_i \subseteq E$. Given a minimum support threshold *minsup*, a set of edges $E' \subseteq E$ is called a frequent edgeset if it appear in at least *minsup* graphs in the relation graph set. Let $S(E')$ denote the set of graph ids of the graphs that contain E'. The support of an edgeset E' is $|S(E')|$.

Because of the anti-monotone property of the support of an edgeset, all subsets of a frequent edgeset are frequent. The frequent edgesets would have large overlap between the patterns and the number of these frequent patterns will be large. To overcome this problem, only the maximal frequent patterns are mined. An edgeset is a maximal frequent edgeset if it is frequent and none of its superset is maximal. We employ the GenMax algorithm for mining all maximal frequent edgesets from the graphs [11]. The set of maximal frequent edgesets is defined as $M = \{M_1, M_2, \cdots, M_{|M|}\}$, where each M_i is a maximal frequent edgeset.

Frequent itemset mining method finds exact frequent edgesets, as opposed to approximate frequent edgesets, which may be too strict. It mines all maximal frequent edgesets, resulting in finding too many edgesets which are difficult to analyze. Many edgesets are highly similar with other edgesets, that is, they have high overlap, producing many duplicate modules in the final result.

3.2.2. Biclustering (BiBit)

Here, we briefly describe how biclustering technique is used to mine frequent edgesets from binary edge occurrence matrix. Let $\mathcal{B}$ be a binary edge occurrence matrix that represents a relation graph set $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ with edgeset $E = \{e_1, e_2, \ldots, e_m\}$. Then column j of $\mathcal{B}$ corresponds to a relation graph G_j and row i corresponds to an edge e_i.

A biclustering algorithm can be applied to $\mathcal{B}$ to mine biclusters, that is, submatrices that have high density of ones. A bicluster is an exact bicluster if it contains no noise, i.e., it consists of all ones. A bicluster is an approximate bicluster if it contains noise. Let $\mathcal{B}'$ be a bicluster of $\mathcal{B}$ such that $E' \subseteq E$ is the edgeset that corresponds to the rows of $\mathcal{B}'$ and $\mathcal{G}' \subseteq \mathcal{G}$ is the graph set that corresponds to the columns of $\mathcal{B}'$. If an edge occurrence matrix $\mathcal{B}$ contains the bicluster $\mathcal{B}'$, it means the edgeset E' occurs in the graphs in $\mathcal{G}'$. Given a minimum support threshold *minsup*, if $|\mathcal{G}'| \geq minsup$, then edgeset E' is a frequent edgeset. Therefore, frequent edgesets can be mined from binary edge occurrence matrix by mining biclusters with at least *minsup* columns.

BiBit [26] is a greedy biclustering algorithm for mining exact biclusters from binary matrix. Because BiBit mines exact biclusters, it still has the problem, as with the frequent itemset mining method, that mining condition is too strict. Apart from mining slightly less number of edgesets due to the greedy nature of the algorithm, BiBit method is essentially the same as the frequent itemset mining method in that they both mine exact edgesets and too many edgesets with high similarity among each other.

3.2.3. Biclustering with Noise (BiBitN)

Here, we briefly describe biclustering with noise (BiBitN) method. BiBitN algorithm is an extension of BiBit algorithm that mines approximate biclusters, that is, biclusters that may contain noise. Because it mines approximate biclusters, the discovered edgesets are approximate frequent. This may lead to the discovery of interesting modules not found with the above methods. However,

the algorithm still mines too many edgeset with high similarity. The detailed description of the method is given in Chapter 4.

3.2.4. Biclustering with Post-Processing Summarization

Here, we briefly describe biclustering with post-processing summarization method. This algorithm extends the BiBitN algorithm and returns a summarized set of biclusters. The above methods mine too many similar edgesets, which are computationally expensive to analyze and lead to mining many duplicate modules. To overcome this problem, we summarize the edgesets to return a set of representative edgesets. The summarized set of edgesets is much smaller in size, producing much less duplicate modules. Here we use on-line summarization process to produced the representative set of edgesets. The detailed description of the method is given in Chapter 5.

3.2.5. Biclustering with On-Line Summarization

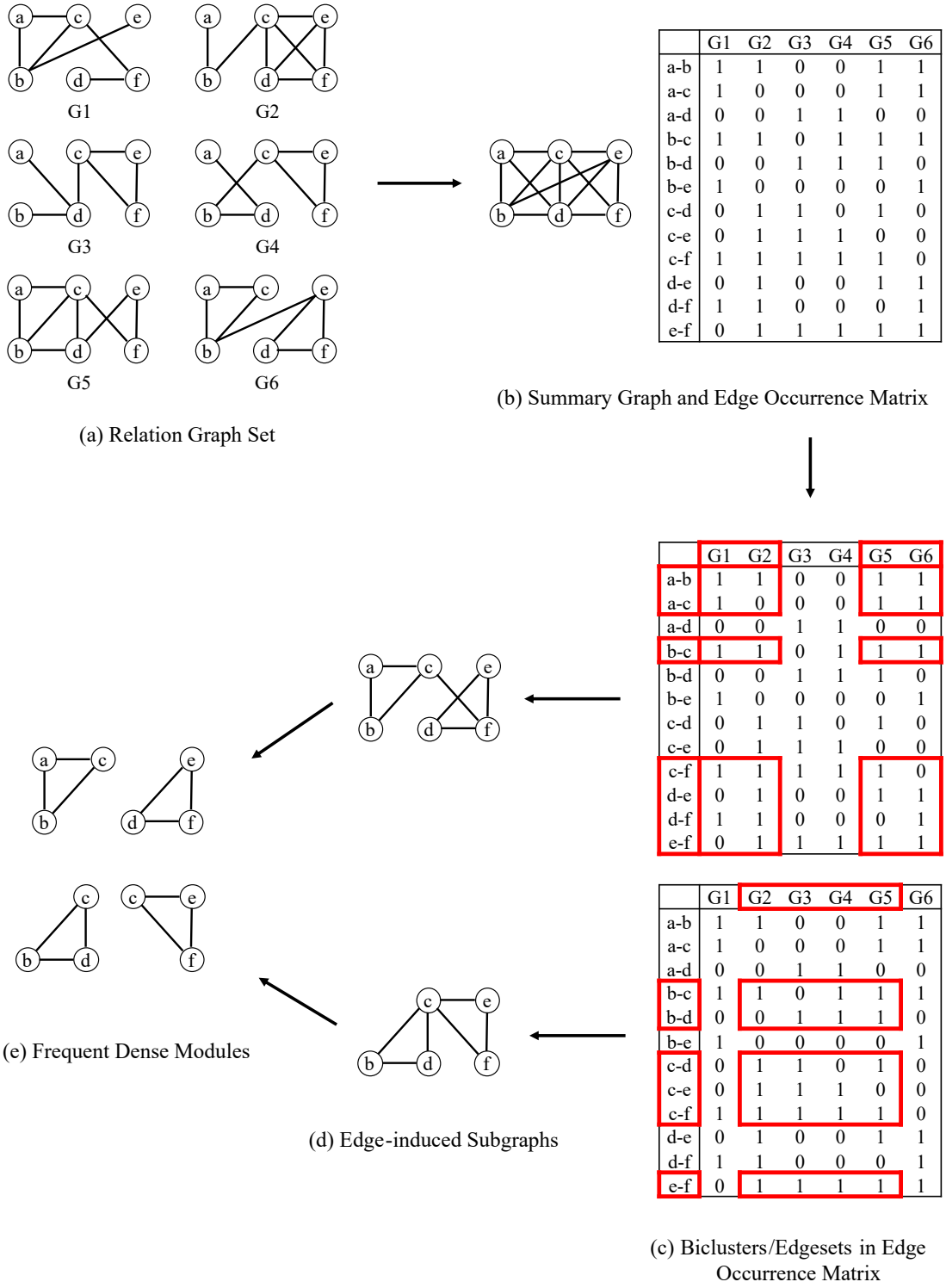
Here, we briefly describe biclustering with on-line summarization method. Similar to the post-processing method, this algorithm extends the BiBitN algorithm and returns a summarized set of biclusters. For pattern summarization method, we use an on-line summarization method instead of post-processing. The post-processing summarization method produces more optimal results but may be computationally expensive, while the on-line summarization is less optimal but less expensive. Additionally, for the on-line summarization, it is guaranteed that no two edgesets in the repersentative set are similar. The detailed description of the method is given in Chapter 6.

3.3. Mining Dense Subgraphs

The second step in our two-step approach is to mine dense modules from the subgraphs extracted from the first step. In this work, we use DME [8] algorithm described in section 2.4.1. For each subgraph mined in the first step, we apply DME algorithm to mine dense modules. The result is a list of frequent dense modules in the original set of relation graphs.

3.4. Mining Frequent Dense Subgraphs

In this section, we summarize our general two-step algorithm for mining frequent dense subgraphs from a relation graph set. The outline of our proposed algorithm is illustrated in Algorithm 4. (Line 1) First the binary edge occurrence matrix is generated from the input relation graph set. (Line 2) Then we mine frequent edgesets from the edge occurrence matrix for a given support threshold and, if we allow noise, the noise ratio. Different methods for mining edgesets can be used in this step, including the five methods discussed earlier, which are frequent itemset mining



	G1	G2	G3	G4	G5	G6
a-b	1	1	0	0	1	1
a-c	1	0	0	0	1	1
a-d	0	0	1	1	0	0
b-c	1	1	0	1	1	1
b-d	0	0	1	1	1	0
b-e	1	0	0	0	0	1
c-d	0	1	1	0	1	0
c-e	0	1	1	1	0	0
c-f	1	1	1	1	1	0
d-e	0	1	0	0	1	1
d-f	1	1	0	0	0	1
e-f	0	1	1	1	1	1

Figure 3.4. Steps in mining frequent dense subgraphs. (a) Relation graph set. (b) Summary graph and the binary edge occurrence matrix generated from the relation graph set. (c) Biclusters/edgesets mined from the edge occurrence matrix. (d) Subgraphs induced by the edgesets. (e) Dense modules mined from edge-induced subgraphs.

Algorithm 4 Mining Frequent Dense Subgraphs

Input:
$\mathcal{G}$: relation graph set
α: minimum support threshold
θ: module density threshold
r: noise ratio
Output:
$\mathcal{X}$: list of frequent dense subgraphs

1. $M = edgeOccurrenceMatrix(\mathcal{G})$
2. $\mathcal{E} = frequentEdgesets(M, \alpha, r)$ ◁ mine frequent edgesets
3. **for** each edgeset E in $\mathcal{E}$ **do**
4. $G = edgeInducedSubgraph(E)$
5. $D = DME(G, \theta)$ ◁ mine dense modules
6. $\mathcal{X} = \mathcal{X} \cup D$
7. **end for**

(GenMax [11]), biclustering (BiBit [26]), biclustering with noise (BiBitN), biclustering with post-processing summarization, and biclustering with on-line summarization. (Line 4) For each edgeset found, we generate an edge-induced subgraph. These subgraphs are frequent but not necessarily dense. (Line 5) Finally, we mine dense modules the edge-induced subgraphs using DME [8]. Note that since the subgraphs of a frequent subgraph are themselves frequent, the output dense modules are frequent. The final subgraphs are exact for frequent itemset mining and biclustering methods, whereas they are approximate for biclustering with noise (BiBitN) method and its extension methods (post-processing and on-line summarizations).

Figure 3.4 shows an outline of the steps in the algorithm. (a) is the original relation graph set from which we mine frequent dense subgraphs. In step (b), we generate the summary graph and the binary edge occurrence matrix from the relation graph set. In step (c), we mine biclusters/edgesets from the edge occurrence matrix, and one of the four methods discussed earlier can be used. The frequent itemset mining and biclustering methods would produce exact biclusters, while the biclustering with noise (BiBitN) method and its extension methods would produce approximate biclusters, as is the case in this example. In step (d), we generate subgraphs induced by the edgesets in (c). Finally in step (e), we mine dense modules from the edge-induced subgraphs in (d). Since the edge-induced subgraphs in (d) are frequent, the final dense modules in (e) are also frequent.

4. BICLUSTERING WITH NOISE (BIBITN)

In this chapter, we present a detailed description of the biclustering with noise method and the associated BiBitN algorithm, followed by the experimental results.

4.1. Biclustering with Noise

As explained earlier, the first two methods described in section 3.2, frequent itemset mining and biclustering, mine exact edgesets. This means the edgesets cannot contain any noise, which may be too strict. Due to the complex procedure of microarray experiments, gene expression data often contains a lot of noise. Because of the noise, there may be some missing coexpression links that have biological significant in reality. Relaxing the condition to mine approximate frequent edgesets, that is, allowing edgesets to contain noise, may help discover biologically significant modules that are otherwise missed.

4.2. BiBitN Algorithm

BiBitN (BiBit with Noise) algorithm is an extension of the BiBit [26] algorithm. Although BiBit is relatively efficient, it is only able to mine exact biclusters, and we extend the algorithm to mine approximate biclusters. The BiBitN algorithm is shown in Algorithm 5. It is mostly the same as the BiBit algorithm shown in Algorithm 2. The main modification is that BiBitN algorithm takes an extra parameter r, which represents the ratio of the maximum allowed noise, and takes it into account when adding rows to seeds. We add a row to a seed if the row matches the seed within the noise limit, even if the row does not match the seed exactly. The maximum allowed number of violating bits per row is determined by the noise ratio r and the size of the particular seed. More precisely, for a given noise ratio r, a row q is considered a valid extension of the seed $S(i, j)$, if $|S(q) \cap S(i, j)|/|S(i, j)| \geq 1 - r$, as shown in line 6 of the algorithm. Consequently, the BiBitN algorithm mines approximate biclusters with noise ratio of at most r.

Figure 4.1 illustrates how a pattern is extended in BiBitN algorithm. Rows i and j produce the seed $S(i, j) = \{2, 3, 7, 9\}$. For the noise ratio $r = 0.25$, both rows k and l are added to the pattern because $|S(k) \cap S(i, j)|/|S(i, j)| = 1$ and $|S(l) \cap S(i, j)|/|S(i, j)| = 0.75$. Note that in BiBit algorithm the row l is not added to the pattern because $S(l) \cap S(i, j) \neq S(i, j)$, as illustrated in Figure 2.4. In other words, BiBit algorithm does not allow row l to be added to the pattern because

Algorithm 5 BiBitN Algorithm

Input:
$\mathcal{B}$: $m \times n$ binary matrix; m edges $\times$ n graphs
mnr: minimum number of rows
mnc: minimum number of columns
r: noise ratio
Output:
$\mathcal{X}$: set of final biclusters/frequent edgesets

1. **for** every edge pair (i, j) **do**
2. $S(i, j) = S(i) \cap S(j)$
3. **if** $S(i, j)$ is new and $|S(i, j)| \geq mnc$ **then**
4. $\langle I, S(i, j)\rangle = \langle\{i, j\}, S(i, j)\rangle$
5. **for** every remainder edge, $q \in E \setminus I$ **do**
6. **if** $|S(q) \cap S(i, j)|/|S(i, j)| \geq 1 - r$ **then**
7. $I = I \cup q$
8. **end if**
9. **end for**
10. **if** $|I| \geq mnr$ **then**
11. $\mathcal{X} = \mathcal{X} \cup \langle I, S(i, j)\rangle$
12. **end if**
13. **end if**
14. **end for**

the column 3 of row l is zero, whereas BiBitN algorithm allows it because three out of the four columns $\{2, 3, 7, 9\}$ of row l are one.

In line 3, the seed $S(i, j)$ is considered visited (not new) if it is similar to an already visited seed. We use Jaccard similarity as the similarity measure. Note that the seed pattern $\langle\{i, j\}, S(i, j)\rangle$ must consist of all ones. A notable feature of the algorithm is that it requires noise to be distributed across the rows. This is a desirable feature because it prevents the discovery of biclusters in which all noise is concentrated in few rows, which may lead to the discovery of subgraphs with false edges.

4.3. Experimental Results

To evaluate the effectiveness of the proposed approach, we mined approximate frequent dense modules from real gene coexpression networks for varying parameter values. Moreover, to assess the biological significance of the reported modules, we conducted Gene Ontology enrichment analysis.

4.3.1. Dataset

We used 35 tissue gene coexpression networks constructed by the Genetic Network Analysis Tool [24]. The coexpression networks were inferred from Genotype-Tissue Expression (GTEx)

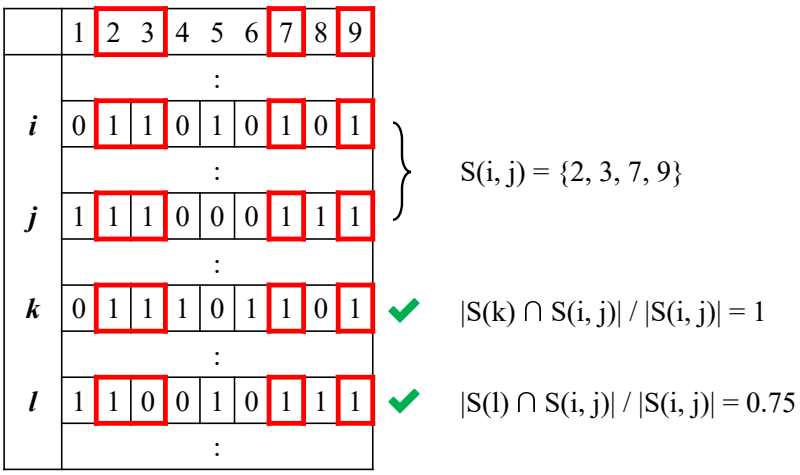


Figure 4.1. Pattern extension in BiBitN for $r = 0.25$. Rows i and j produce seed $S(i, j) = \{2, 3, 7, 9\}$. Both rows k and l are added to the pattern because $|S(k) \cap S(i, j)|/|S(i, j)| = 1$ and $|S(l) \cap S(i, j)|/|S(i, j)| = 0.75$.

data [1]. Each coexpression network is constructed from the gene expression of non-diseased tissue samples. On average there are $14,415$ links in each network among $9,998$ genes. In total, there are $1,548,622$ unique links that appear in at least one network, $4,127$ edges that appear in at least 20 networks, and on average each link appears in 3.28 networks.

4.3.2. Topological Analysis of Frequent Edgesets

We constructed the binary edge occurrence matrix from the 35 gene coexpression networks and ran the BiBitN algorithm for support thresholds from 20 to 24, and for noise thresholds 0, 0.1, 0.2, and 0.3. We used 0.7 as the seed similarity threshold. Table 4.1 shows the topological properties of the frequent edgesets for various support and noise thresholds. M is the number of frequent edgesets and $\overline{E}$ is the average number of edges in each frequent edgeset. Figure 4.2 shows how the number of frequent edgesets varies for different parameter values. We can see that the number of frequent edgesets decreases with increasing support threshold. We can also see that the number increases slightly with increasing noise, but not by much. Figure 4.3 shows how the average edgeset size varies for different parameter values. We get noticeably larger frequent edgesets for larger noise thresholds. This is expected because increasing noise threshold allows more edges to be added to the patterns.

Notice that the edgeset size is expected to be smaller for larger support threshold, but it is not shown in the results. This is because when calculating the allowed number of zeros per row in

[1] `https://www.gtexportal.org/`

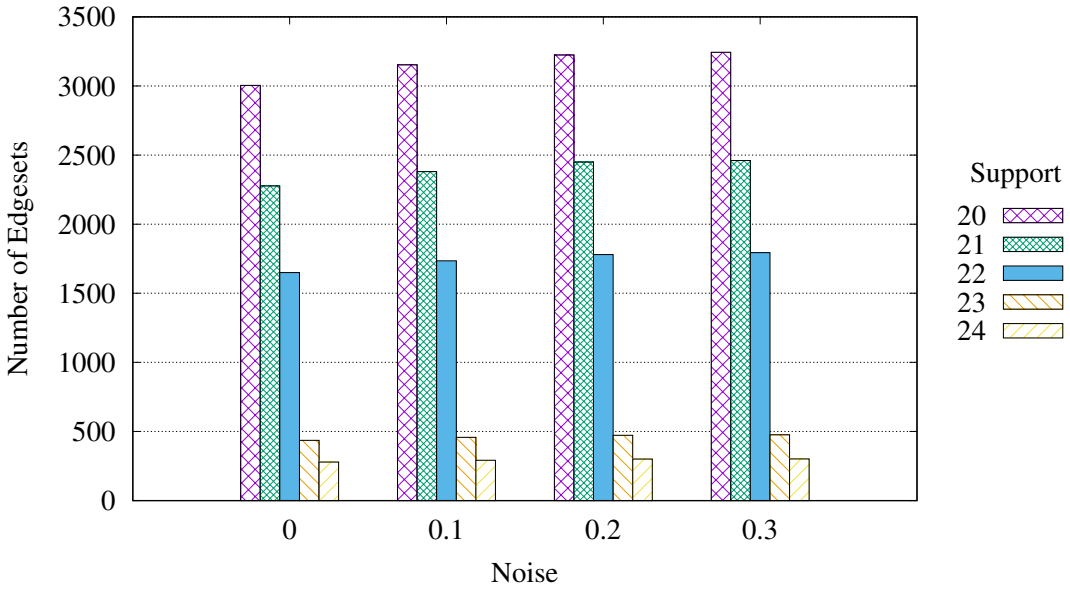


Figure 4.2. Number of frequent edgesets for varying noise thresholds for BiBitN approach.

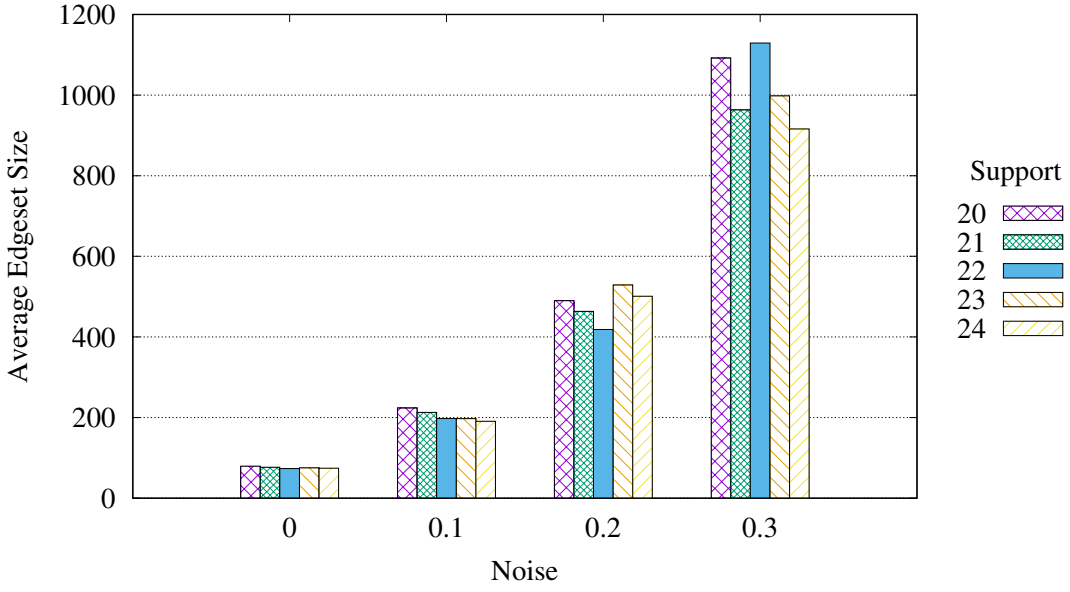


Figure 4.3. Average edgeset size for varying noise thresholds for BiBitN approach.

Table 4.1. Topological properties of the frequent edgesets for BiBitN approach. M is the number of frequent edgesets and $\overline{E}$ is the average number of edges in each frequent edgeset.

noise	0		0.1		0.2		0.3	
minsup	M	$\overline{E}$	M	$\overline{E}$	M	$\overline{E}$	M	$\overline{E}$
20	3 K	79.6	3.2 K	224	3.2 K	490	3.2 K	1.1 K
21	2.3 K	76.7	2.4 K	212.8	2.4 K	463.4	2.5 K	963.4
22	1.6 K	73.4	1.7 K	197.7	1.8 K	418.6	1.8 K	1.1 K
23	436	75.4	458	197.5	472	528.9	476	998.2
24	279	74.5	292	190.9	300	501	301	916

each bicluster, we round the number to the nearest integer. More precisely, $n = int(d * r)$, where n is the allowed number of zeros per row, d is the number of columns in the bicluster, r is the noise threshold, and $int(x)$ denotes the nearest integer to x. For example for noise 0.2, the average edgeset size is smaller for support 22 than for 23, because a bicluster with 22 columns allowes 4 zeros while one with 23 columns allows 5 zeros.

4.3.3. Topological Analysis of Frequent Dense Modules

We mined dense modules from the subgraphs induced by the frequent edgesets. We used density thresholds 0.5 and 0.6, and only modules with at least four nodes were considered. The topological properties of the frequent dense modules for various parameter values are shown in Table 4.2. M' denotes the number of approximate frequent edgesets that have at least one dense module for the specified density threshold, $\overline{DM}$ denotes the average number of dense modules in edge-induced subgraph of each edgeset, and $\overline{V'}$ denotes the average size of the dense modules. As the minimum support is increased, the number of approximate frequent edgesets decreases, and the average number of dense modules also decreases. For larger density thresholds, the average size of the dense modules decreases. As the noise threshold is increased we get larger dense modules.

Because many frequent edgesets have large overlap, there are many duplicate dense modules. Table 4.3 shows the number of all frequent dense modules, unique frequent dense modules, and the percentage of the unique frequent dense modules. We can see that the percentage of the unique dense modules increases with increasing support threshold.

Table 4.2. Topological properties of the frequent dense modules for BiBitN approach. M' is the number of frequent edgesets that have at least one dense module, $\overline{DM}$ is the average number of dense modules in each edge-induced subgraph, and $\overline{V'}$ is the average size of the dense modules.

noise		0			0.1			0.2		
minsup	density	M'	$\overline{DM}$	$\overline{V'}$	M'	$\overline{DM}$	$\overline{V'}$	M'	$\overline{DM}$	$\overline{V'}$
20	0.5	1.7 K	10.6	4	2.9 K	46.8	4.1	3.1 K	219.8	4.2
	0.6	26	1	4	895	2.9	4.2	2.1 K	12	4.4
21	0.5	1.3 K	9.9	4	2.2 K	41.7	4.1	2.4 K	198.4	4.2
	0.6	16	1.1	4	594	2.6	4.1	1.6 K	10.5	4.3
22	0.5	899	9.6	4	1.6 K	36	4.1	1.7 K	165.1	4.2
	0.6	8	1	4	362	2.5	4.1	1.1 K	9	4.3
23	0.5	248	9.3	4	426	35.3	4.1	460	249.4	4.2
	0.6	1	1	4	107	2.4	4.1	342	13.9	4.3
24	0.5	167	8.4	4	274	32.6	4.1	294	224.7	4.2
	0.6	2	1	4	64	2.2	4.1	216	12.6	4.3

4.3.4. Gene Ontology Enrichment Analysis

To assess the biological significance of the reported modules, we conducted Gene Ontology enrichment analysis of the reported unique frequent dense modules. The analysis shows that the modules are enriched with KEGG pathways and molecular functions. A frequent dense module is enriched if it overlaps with the geneset of a known molecular signature. We used the overrepresentation of genes with a specific annotation in a gene set using the hybergeometric test (with $pvalue = 0.01$). For biological terms, we used the KEGG pathway database (186 sets covering $5,241$ genes) and the GO Molecular Function Ontology ($1,645$ sets covering $15,599$ genes). Table 4.4 shows the percentage of frequent dense modules that are biologically enriched. E_{MF} and E_{KEGG} denote the percent enriched in molecular functions and KEGG pathways respectively. As shown in the results, frequent dense modules with smaller noise ratios have higher enrichment. Enrichment with GO molecular functions is higher than KEGG pathways since there are much more molecular functions and they cover more genes. Table 4.5 shows the top biological signatures that were enriched the most in the reported modules for $sup = 20$, $noise = 0.1$, and $density = 0.5$.

Table 4.3. Number of all frequent dense modules, unique frequent dense modules, and the percentage of the unique frequent dense modules for BiBitN approach.

noise		0			0.1			0.2		
minsup	density	all	unique	%	all	unique	%	all	unique	%
20	0.5	18.3 K	259	1.4	137.9 K	2 K	1.5	687.4 K	12.2 K	1.8
	0.6	27	6	22.2	2.6 K	115	4.4	25.8 K	922	3.6
21	0.5	12.7 K	218	1.7	93.1 K	1.6 K	1.7	468.9 K	8.6 K	1.8
	0.6	17	6	35.3	1.6 K	79	5.1	17 K	618	3.6
22	0.5	8.6 K	167	1.9	58.5 K	1.2 K	2	285 K	6.1 K	2.1
	0.6	8	2	25	904	56	6.2	9.9 K	449	4.5
23	0.5	2.3 K	103	4.5	15 K	729	4.8	114.7 K	5.6 K	4.9
	0.6	1	1	100	252	28	11.1	4.7 K	424	8.9
24	0.5	1.4 K	71	5.1	8.9 K	523	5.9	66.1 K	3.7 K	5.7
	0.6	2	2	100	139	18	12.9	2.7 K	289	10.6

Table 4.4. GO term enrichment analysis for frequent dense modules for BiBitN approach. E_{MF} and E_{KEGG} denote the percent enriched in molecular functions and KEGG pathways respectively.

noise		0		0.1		0.2	
minsup	density	E_{MF}	E_{KEGG}	E_{MF}	E_{KEGG}	E_{MF}	E_{KEGG}
20	0.5	83.4	63.7	75.1	63	59.9	49.9
	0.6	100	33.3	84.3	53	75.5	54.2
21	0.5	84.4	64.2	77.1	63.9	62.8	52.4
	0.6	100	33.3	84.8	59.5	77.5	52.4
22	0.5	86.8	65.9	77.5	62.5	65.8	54.4
	0.6	100	0	91.1	57.1	78.6	50.3
23	0.5	84.5	61.2	82	63.8	64.3	53.2
	0.6	100	0	96.4	50	77.1	48.6
24	0.5	81.7	50.7	81.6	66.7	68.4	55.5
	0.6	100	0	94.4	61.1	80.6	47.8

Table 4.5. Top enriched biological signatures in the reported modules for $sup = 20$, $noise = 0.1$, and $density = 0.5$ for BiBitN approach.

GO Molecular Function	Count
STRUCTURAL CONSTITUENT OF RIBOSOME	832
RRNA BINDING	214
5S RRNA BINDING	144
ANTIGEN BINDING	135
ELECTRON TRANSFER ACTIVITY	127
IMMUNOGLOBULIN RECEPTOR BINDING	123
OXIDOREDUCTASE ACTIVITY ACTING ON NAD P H	99
NADH DEHYDROGENASE ACTIVITY	95
OXIDOREDUCTASE ACTIVITY ACTING ON A HEME GROUP OF DONORS	91

KEGG pathway	Count
RIBOSOME	834
HUNTINGTONS DISEASE	235
PARKINSONS DISEASE	216
OXIDATIVE PHOSPHORYLATION	215
ALZHEIMERS DISEASE	213
CARDIAC MUSCLE CONTRACTION	109
AUTOIMMUNE THYROID DISEASE	32
VIRAL MYOCARDITIS	23
MAPK SIGNALING PATHWAY	22
AMINOACYL TRNA BIOSYNTHESIS	20

5. BICLUSTERING WITH POST-PROCESSING SUMMARIZATION

In this chapter, we present a detailed description of the biclustering with post-processing summarization method, followed by the experimental results.

5.1. Mining Representative Frequent Edgesets

The BiBitN algorithm reports huge number of frequent connected subgraphs, especially for low support thresholds. This makes the downstream analysis of these frequent subgraphs very difficult. Moreover, when these subgraphs are used for graph clustering and classification, this large number of subgraphs can have a negative impact on the accuracy of the methods. Moreover, many edgesets have large overlap with each other, producing many duplicate modules in the final set of frequent dense modules. To overcome this problem, we need to report a summarized set of edgesets. This set would be representative of the reported edgesets such that every edgeset not included in the representative set has at least one similar edgeset in the representative set. The definition of the set of representative edgesets follows:

Set of Representative Edgesets: Given a set of edgeset patterns P and an edgeset similarity threshold s, a subset $P' \subseteq P$ is a set of representative edgesets if for every edgeset $E \in P \setminus P'$, there exists an edgeset $E' \in P'$ such that $sim(E, E') \geq s$, where $sim(E, E')$ is the similarity between the two sets. Each edgeset in P is either in P' or is similar to an edgeset in P'.

In order to reduce the number of reported patterns and decrease the overlap between the reported patterns, we use a post-processing data summarization method to mine a set of representative frequent edgesets. In the proposed post-processing data summarization method, all frequent edgesets are first mined and then a subset of these frequent edgesets is chosen such that every edgeset not in the set has at least one similar edgeset in the set. The summarization method uses the concept of similarity graph and dominating set to choose the representative frequent edgesets [10].

5.1.1. Similarity Measure

We use the Jaccard similarity coefficient to measure the similarity between edgesets. The Jaccard similarity coefficient between two sets is defined as the cardinality of the intersection of the two sets divided by the cardinality of the union of the two sets. More precisely, the Jaccard similarity coefficient of the two sets A and B is

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The similarity score ranges between 0 and 1. Roughly, it is the measure of the degree of overlap between the two sets, with 0 indicating no similarity and 1 indicating identical sets. In general, the size of the representative set is smaller for lower value of edgeset similarity threshold. For the special case when the similarity threshold is set to 1, the set of representative frequent edgesets is the same as the set of all frequent edgesets. And for the special case when the similarity threshold is set to 0, the first encountered frequent edgeset is the only pattern in the set, as it is 'similar' to all other edgesets.

5.1.2. Similarity Graph

We first mine all the frequent edgesets using the BiBitN algorithm. Once we mine the set of all frequent edgesets, we construct the similarity graph to represent the similarities between the edgesets. In the similarity graph, each node corresponds to an edgeset, and there is an edge between two nodes if the similarity between the two corresponding edgesets exceeds a user-defined similarity threshold. More formally, given a set of m frequent edgeset patterns $P = \{P_1, P_2, \cdots, P_m\}$ and a user-defined similarity threshold s, the similarity graph $G_P(V_P, E_P)$ is a graph such that each node $v_i \in V_P$ corresponds to pattern $P_i \in P$ and there is an edge $(v_i, v_j) \in E_P$ if $sim(P_i, P_j) \geq s$, where $sim(P_i, P_j)$ is the similarity between patterns P_i and P_j. Figure 5.1 (b) shows the similarity graph constructed from the set of edgeset patterns in (a) with similarity threshold 0.5. For example, the similarity graph in (b) has the edge (P_1, P_2) because the similarity between edgesets P_1 and P_2 is 0.5.

5.1.3. Dominating Set

A dominating set of a graph $G(V, E)$ is a subset $S \subseteq V$ such that every node not in S is connected to at least one node in S. A minimum dominating set is the smallest such set. A graph

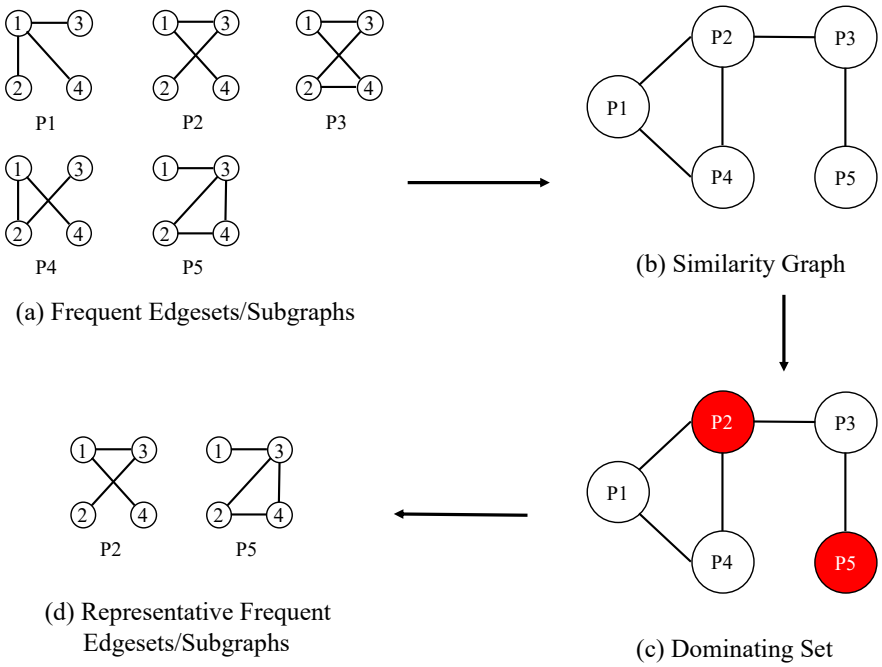


Figure 5.1. Steps in mining representative frequent subgraphs from set of all frequent subgraphs: (a) Set of all frequent edgesets/subgraphs; (b) Similarity graph for the set in (a) with similarity threshold 0.5 (Jaccard similarity coefficient); (c) Dominating set of the similarity graph in (b); (d) Set of representative frequent edgesets/subgraphs

can have multiple minimum dominating sets. Since the similarity graph for the set of all frequent edgesets represents edgeset similarities, a minimum dominating set of the similarity graph is the smallest node set which corresponds to the set of representative frequent edgesets. Figure 5.1 (c) shows a minimum dominating set for the the similarity graph in (b). The corresponding representative frequent edgesets are shown in (d). For the similarity graph $G_P(V_P, E_P)$, the goal is to find a subset of vertices (patterns) $S_P \subseteq V_P$ that dominates all the remaining vertices (patterns). The problem of finding a minimum dominating set of a graph is NP-hard. There are linear reductions between the set cover problem, a well-known NP-hard problem, and the minimum dominating set problem [4]. Therefore, we employ an approximation greedy algorithm whose solution is optimal up to a certain factor. The greedy algorithm starts with an empty set, $S = \emptyset$, and adds vertices to S until S is a dominating set of the graph. The most common greedy algorithm is to select the vertex that has the maximum number of neighbors that are not dominated. The number of undominated vertices that a vertex v dominates is denoted by $du(v)$. Initially each vertex dominates itself and its neighbors. So the vertex with the largest degree is chosen as the first vertex to add

Algorithm 6 BiBitN with Post-Processing Summarization

Input:
$\mathcal{B}$: $m \times n$ binary matrix; m edges $\times$ n graphs
mnr: minimum number of rows
mnc: minimum number of columns
r: noise ratio
s: edgeset similarity threshold
Output:
$\mathcal{X}$: set of representative frequent edgesets

1. $\mathcal{X}' = BiBitN(\mathcal{B}, mnr, mnc, r)$
2. $\mathcal{G} = similarityGraph(\mathcal{X}', s)$
3. $\mathcal{X} = minimumDominatingSet(\mathcal{G})$

to S. Next the du score is updated for all vertices and the algorithm selects the vertex with the largest du score. If there are multiple vertices with the largest score, a vertex is chosen randomly. This process is repeated until all vertices are dominated. For the similarity graph in Figure 5.1 (b), the greedy algorithm selects P_2 as the first vertex in the dominating set S since P_2 dominates four vertices including itself. After updating the du score, both P_3, and P_5 have the same score of 1. The algorithm chooses one of them randomly. Note that P_3 is still a candidate to be added to the dominating set even though it is dominated. The algorithm then selects P_5 and terminates since all vertices are dominated now. The final dominating set is $S = \{P_2, P_5\}$, indicating that the corresponding patterns are the representative frequent edgesets.

5.2. Algorithm

The algorithm extends BiBitN algorithm and incorporates the post-processing summarization method to reduce the number of reported frequent edgesets. It is illustrated in Algorithm 6. It takes an additional input parameter s, which is the edgeset similarity threshold. We first run the BiBitN algorithm on the given parameters $\mathcal{B}, mnr, mnc$, and r to mine set of all frequent edgesets $\mathcal{X}'$. Instead of returning the set of all frequent edgesets, we mine the set of representative edgesets using the post-processing summarization method in line 2-3. The similarity graph $\mathcal{G}$ is constructed from the set of all frequent patterns $\mathcal{X}'$ and the similarity threshold s. Then the minimum dominating set of $\mathcal{G}$ is computed. The frequent edgesets in the set $\mathcal{X}'$ that correspond to the elements of the minimum dominating set are returned as the set of representative frequent edgesets $\mathcal{X}$.

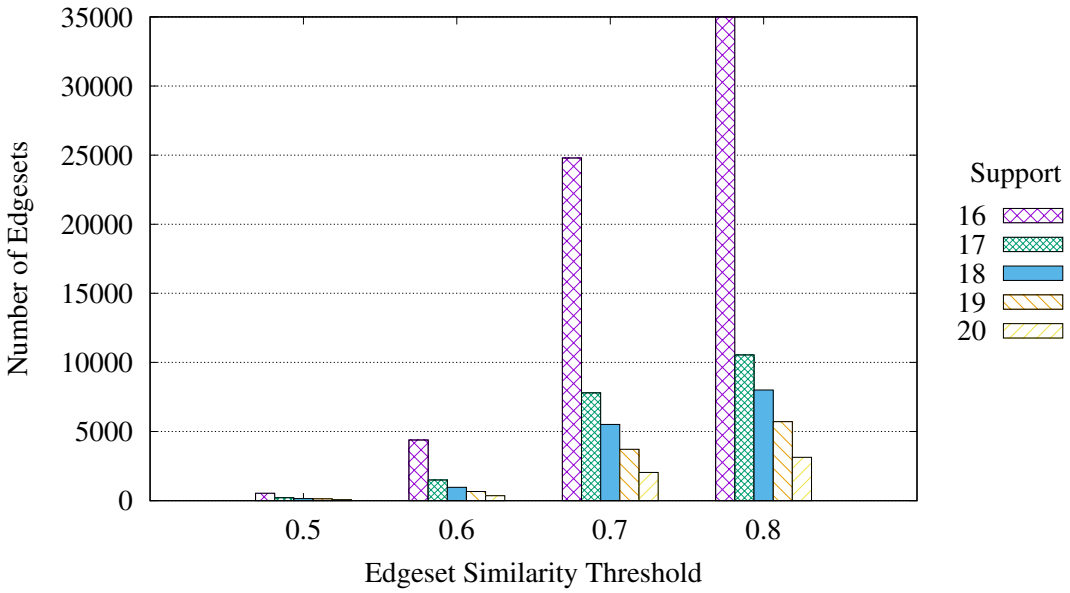


Figure 5.2. Number of frequent edgesets for varying edgeset similarity thresholds for biclustering with post-processing summarization approach.

5.3. Experimental Results

To evaluate the effect of the post-processing data summarization, we mined approximate frequent edgesets for various similarity threshold values. Moreover, to evaluate the overall effectiveness of the proposed approach, we mined approximate frequent dense modules from the 35 gene coexpression network described in section 4.3.1. We also conducted Gene Ontology enrichment analysis to assess the biological significance of the reported modules.

5.3.1. Effect of Edgeset Similarity Threshold

To see the effect of the data summarization, we ran the post-processing summarization algorithm on the binary edge occurrence matrix constructed from the 35 gene coexpression networks, for support threshold values from 16 to 20, noise threshold 0.1, and edgeset similarity threshold values 0.5, 0.6, 0.7, and 0.8. Figure 5.2 shows hows the number of frequent edgesets varies for different edgeset similarity threshold values. We can see that the number of frequent edgesets decreases with increasing support threshold and increases with increasing edgeset similarity threshold. This is expected because less number of representative edgesets is needed for lower similarity threshold. Figure 5.3 shows how the average edgeset size varies for different edgeset similarity threshold values. We see that the average edgeset size increases with increasing edgeset similarity threshold.

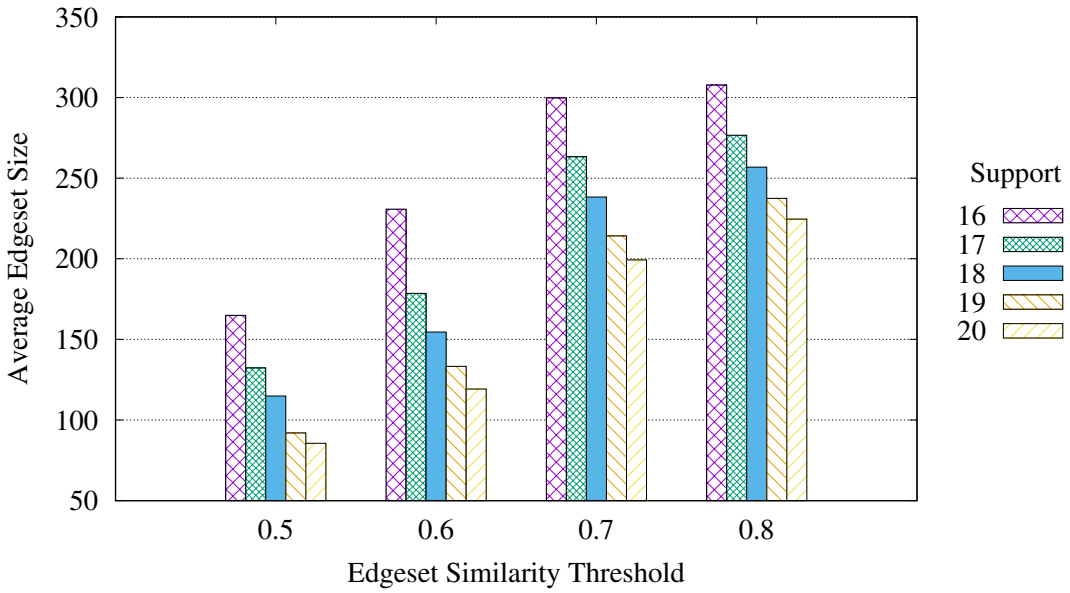


Figure 5.3. Average edgeset size for varying edgeset similarity thresholds for biclustering with post-processing summarization approach.

Table 5.1. Comparison of the number of edgesets for support 20 for varying similarity thresholds

noise	0	0.1	0.2	0.3
Without summarization	3,004	3,153	3,224	3,244
With summarization ($s = 0.5$)	113	73	62	81
With summarization ($s = 0.6$)	407	360	341	453
With summarization ($s = 0.7$)	1,226	2,044	2,310	2,894
With summarization ($s = 0.8$)	2,693	3,131	3,220	3,243

To evaluate the effect of the post-processing summarization for frequent edgesets, we mined the frequent frequent edgesets for $minsup = 20$, and used edgeset similarity thresholds 0.5 to 0.8 for mining representative frequent edgesets. The number of reported frequent edgesets for various edgeset similarity thresholds is shown in Table 5.1. It shows that the number of representative frequent edgesets increases with increasing similarity threshold. For a large similarity threshold, fewer edgesets are similar to each other and therefore the number of representative patterns is larger. For a small similarity threshold, less number of patterns is needed to represent the entire set.

5.3.2. Topological Analysis of Frequent Edgesets

We ran the post-processing summarization algorithm on the binary edge occurrence matrix for support threshold values from 16 to 20, noise threshold values from 0 to 0.3, and edgeset similarity threshold 0.6. Table 5.2 shows the topological properties of the frequent edgesets for varying parameter values. M is the number of frequent edgesets and $\overline{E}$ is the average number of edges in each frequent edgeset. Figure 5.4 shows how the number of frequent edgesets varies for different noise threshold values. Figure 5.5 shows how the average edgeset size varies for different noise threshold values. We see that the average edgeset size increases with increasing noise threshold, as expected.

We can see that compared to the BiBitN approach, this approach produces much smaller number of frequent edgesets, especially for low values of edgeset similarity threshold. This makes analysis with lower values of support threshold more feasible.

Table 5.2. Topological properties of the frequent edgesets for biclustering with post-processing summarization approach with edgeset similarity threshold 0.6. M is the number of frequent edgesets and $\overline{E}$ is the average number of edges in each frequent edgeset.

noise	0		0.1		0.2		0.3	
minsup	M	$\overline{E}$	M	$\overline{E}$	M	$\overline{E}$	M	$\overline{E}$
16	3.3 K	45	4.4 K	230.8	4.2 K	405.4	6.9 K	1.1 K
17	1.3 K	40.7	1.5 K	178.5	1.6 K	384	2.1 K	857.6
18	945	36.2	963	154.6	1.1 K	411.8	1.5 K	774.9
19	697	32.8	660	133.3	604	349.8	980	833.3
20	407	32.7	360	119.2	341	287.5	453	711.4

5.3.3. Topological Analysis of Frequent Dense Modules

We mined dense modules from the subgraphs induced by the frequent edgesets, using density thresholds 0.5 and 0.6, and only modules of size four or larger were considered. Table 5.3 shows the topological properties of the frequent dense modules for varying parameter values and edgeset similarity threshold 0.6. M' denotes the number of approximate frequent edgesets that have at least one dense module for the specified density threshold, $\overline{DM}$ denotes the average number of dense modules in edge-induced subgraph of each edgeset, and $\overline{V'}$ denotes the average size of the

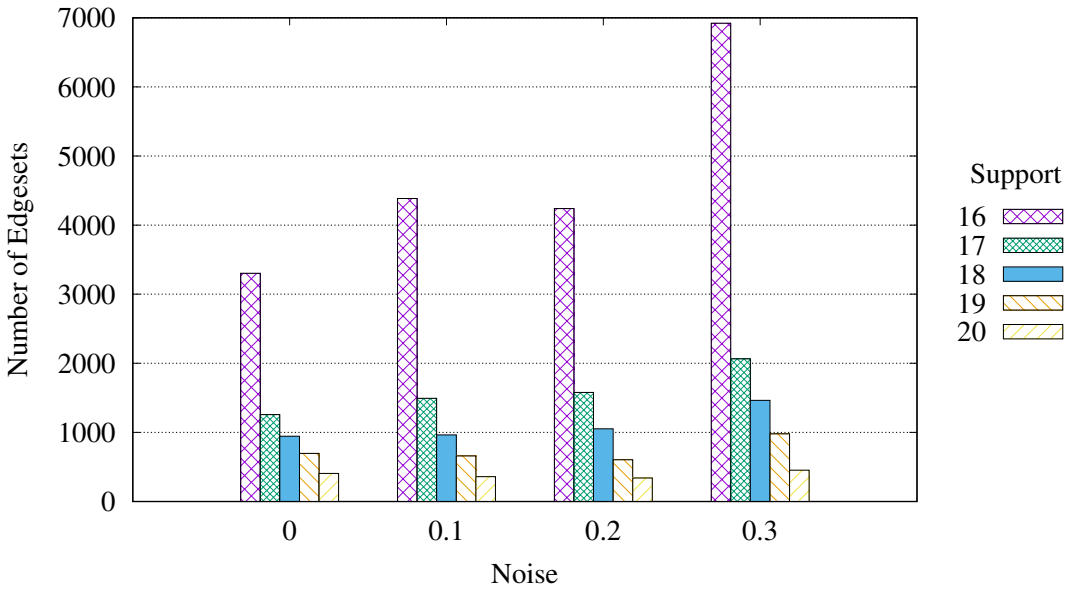


Figure 5.4. Number of frequent edgesets for varying noise thresholds for biclustering with post-processing summarization approach with edgeset similarity threshold 0.6.

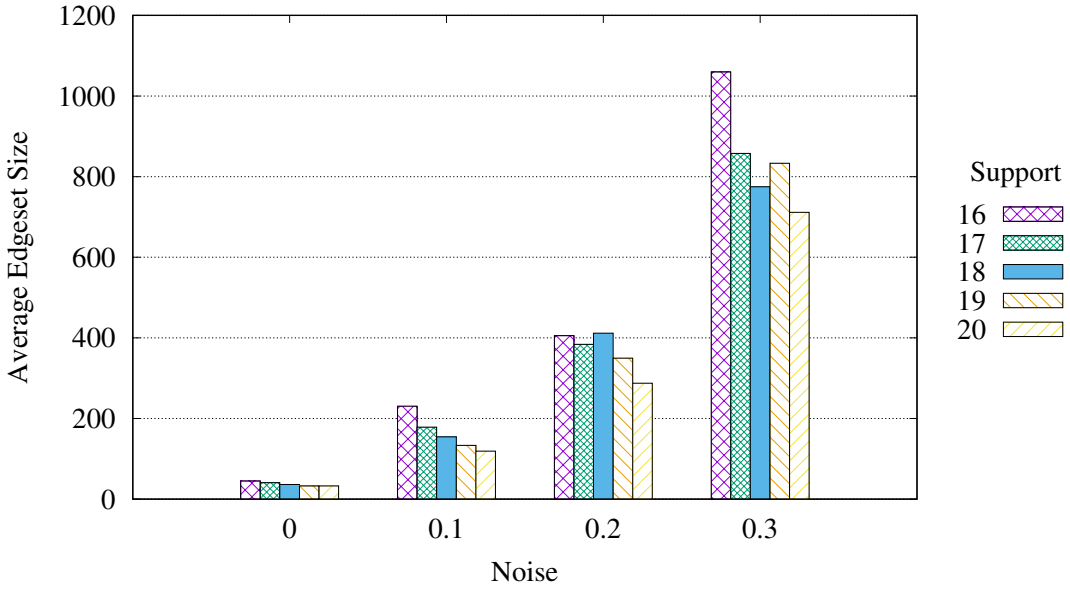


Figure 5.5. Average edgeset size for varying noise thresholds for biclustering with post-processing summarization approach with edgeset similarity threshold 0.6.

dense modules. We can see that the number of edgesets with at least one dense module decreases as the support threshold is increased.

Table 5.4 shows the number of all frequent dense modules, unique frequent dense modules, and the percentage of the unique frequent dense modules. As with the BiBitN approach, the percentage of the unique dense modules increases with increasing support threshold. We can see that the percentages of the unique dense modules are much higher compared to the BiBitN results, which means there are much smaller number of duplicate modules.

Table 5.3. Topological properties of the frequent dense modules for biclustering with post-processing summarization approach with edgeset similarity threshold 0.6. M' is the number of frequent edgesets that have at least one dense module, $\overline{DM}$ is the average number of dense modules in each edge-induced subgraph, and $\overline{V'}$ is the average size of the dense modules.

noise		0			0.1			0.2		
minsup	density	M'	$\overline{DM}$	$\overline{V'}$	M'	$\overline{DM}$	$\overline{V'}$	M'	$\overline{DM}$	$\overline{V'}$
17	0.5	250	10.2	4	1.3 K	43.6	4.2	1.4 K	181.8	4.3
	0.6	9	1	4	312	4.4	4.2	751	15.3	4.4
18	0.5	164	8.8	4	780	35.5	4.1	949	211.7	4.3
	0.6	1	1	4	162	4	4.1	509	17.6	4.4
19	0.5	107	7.1	4	515	28.1	4.1	526	171.1	4.3
	0.6	1	1	4	79	3.3	4.2	262	15.4	4.4
20	0.5	66	7.4	4	269	23.5	4.1	291	130.4	4.2
	0.6	1	1	4	36	2.6	4	128	11.5	4.3

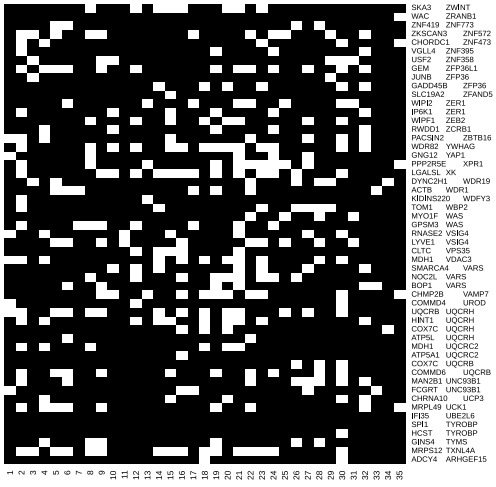
5.3.4. Gene Ontology Enrichment Analysis

To assess the biological significance of the reported modules, we conducted Gene Ontology enrichment analysis the same way as described in section 4.3.4. Table 5.5 shows the percentage of frequent dense modules that are biologically enriched. E_{MF} and E_{KEGG} denote the percent enriched in molecular functions and KEGG pathways respectively. As with the BiBitN results, modules with smaller noise ratio have higher enrichment, and the enrichment with GO molecular functions is higher than the enrichment with KEGG pathways. Table 5.6 shows the top biological signatures that were enriched the most in the reported modules for $sup = 17$, $noise = 0.1$, and $density = 0.5$.

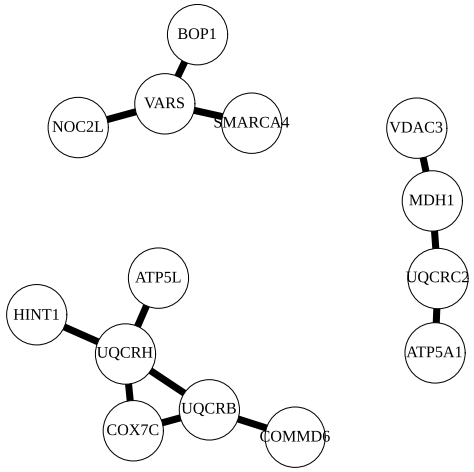
Table 5.4. Number of all frequent dense modules, unique frequent dense modules, and the percentage of the unique frequent dense modules for biclustering with post-processing summarization approach with similarity threshold 0.6.

noise		0			0.1			0.2		
minsup	density	all	unique	%	all	unique	%	all	unique	%
17	0.5	2.5 K	313	12.3	55 K	4.1 K	7.5	262.1 K	20.6 K	7.9
	0.6	9	3	33.3	1.4 K	242	17.6	11.5 K	1.6 K	13.5
18	0.5	1.4 K	165	11.4	27.7 K	2.5 K	9.1	200.9 K	19.8 K	9.9
	0.6	1	1	100	641	146	22.8	8.9 K	1.5 K	16.6
19	0.5	765	96	12.5	14.5 K	1.5 K	10.3	90 K	11.6 K	12.8
	0.6	1	1	100	263	72	27.4	4 K	814	20.2
20	0.5	491	71	14.5	6.3 K	870	13.8	37.9 K	6.1 K	16
	0.6	1	1	100	92	28	30.4	1.5 K	407	27.6

Figure 5.6 shows an example of a frequent edgeset for $sup = 19$, $noise = 0.2$. (a) show the submatrix of the binary edge occurrence matrix which represents the edge occurrences in the frequent edgeset in the 35 networks. Each row corresponds to an edge in the edgeset, and each column corresponds to a gene coexpression network. (b) shows the dense modules mined from the subgraph induced by the edgeset for density threshold 0.5. Nodes are labeled by their corresponding gene identifiers. The genes in this representative approximate edgeset are enriched with five KEGG pathways; Oxidative Phosphorylation, Cardiac Muscle Contraction, Alzheimers Disease, Parkinsons Disease, and Huntingtons Disease, and two molecular functions; Electron Transfer Activity, and Oxidoreductase Activity.



(a) Submatrix for Frequent Edgeset



(b) Frequent Dense Modules

Figure 5.6. Sample frequent edgeset for $minsup = 19$ and $noise = 0.2$, and dense modules in the edgeset for $density = 0.5$

Table 5.5. GO term enrichment analysis for frequent dense modules for biclustering with post-processing summarization approach with edgeset similarity threshold 0.6. E_{MF} and E_{KEGG} denote the percent enriched in molecular functions and KEGG pathways respectively.

noise		0		0.1		0.2	
minsup	density	E_{MF}	E_{KEGG}	E_{MF}	E_{KEGG}	E_{MF}	E_{KEGG}
17	0.5	81.5	61.3	66.8	56.7	51.7	44.2
	0.6	100	33.3	79.8	59.1	71.5	52.7
18	0.5	82.4	62.4	72.7	60.9	52.3	44.2
	0.6	100	0	79.5	56.8	70.2	51.8
19	0.5	86.5	62.5	75.1	63.1	55.6	47
	0.6	100	0	87.5	62.5	74.4	54.2
20	0.5	80.3	57.7	80.3	67.1	62.7	52.2
	0.6	100	0	89.3	75	78.9	56.5

Table 5.6. Top enriched biological signatures in the reported modules for $sup = 17$, $noise = 0.1$, and $density = 0.5$ for biclustering with post-processing summarization approach with edgeset similarity threshold 0.6.

GO Molecular Function	Count
STRUCTURAL CONSTITUENT OF RIBOSOME	1509
RRNA BINDING	389
5S RRNA BINDING	219
ELECTRON TRANSFER ACTIVITY	187
ANTIGEN BINDING	149
UBIQUITIN PROTEIN TRANSFERASE REGULATOR ACTIVITY	146
OXIDOREDUCTASE ACTIVITY ACTING ON NAD P H	137
IMMUNOGLOBULIN RECEPTOR BINDING	135
NADH DEHYDROGENASE ACTIVITY	132

KEGG pathway	Count
RIBOSOME	1511
HUNTINGTONS DISEASE	368
OXIDATIVE PHOSPHORYLATION	362
PARKINSONS DISEASE	344
ALZHEIMERS DISEASE	337
CARDIAC MUSCLE CONTRACTION	167
AUTOIMMUNE THYROID DISEASE	50
AMINOACYL TRNA BIOSYNTHESIS	46
MAPK SIGNALING PATHWAY	43
LEISHMANIA INFECTION	30

6. BICLUSTERING WITH ON-LINE SUMMARIZATION

In this chapter, we present a detailed description of the biclustering with on-line summarization method, followed by the experimental results.

6.1. Mining Representative Frequent Edgesets

The post-processing approach for mining the set of representative frequent edgeset produces (nearly) optimal results. However, this approach has its drawback. The problem with the post-processing approach is that it is computationally expensive. The construction of the similarity graph requires similarity computation of every pair of patterns, which has $O(n^2)$ time complexity where n is the number of patterns. Thus, this approach is not feasible for low support thresholds, where the number of patterns is large.

Therefore, we propose the on-line summarization method for mining the set of representative frequent edgesets. In the on-line approach, we process the patterns as they are produced. We begin with an empty set of representative patterns. When a pattern is found, we check whether or not it has a similar pattern in the representative set. If there is no similar pattern, we add the pattern to the set of representative patterns. As a result, the final set contains patterns such that every pattern not in the set has least one similar pattern in the set. Moreover, no two patterns in the set are similar. In general, the on-line approach does not produce an optimal result, that is, the set with the minimum number of representative patterns. The result depends on the order in which the patterns are mined. The on-line approach is much more efficient in time complexity.

6.2. Algorithm

The algorithm extends the BiBitN algorithm and incorporates the on-line summarization method to mine the set of representative frequent edgesets. It is illustrated in Algorithm 7. It takes an additional input parameter s, which is the edgeset similarity threshold. In lines 10-12, an addition condition is required to add the edgeset I to the final set of edgesets $\mathcal{X}$. The edgeset I is not added to $\mathcal{X}$ if there is at least one edgeset I' in $\mathcal{X}$ that is similar to i. $sim(I, I')$ is the similarity between I and I'. When the algorithm finishes running, the set $\mathcal{X}$ contains representative edgesets and no two edgesets in $\mathcal{X}$ are similar. For the similarity measure, we use the Jaccard similarity coefficient described in Section 5.1.1.

Algorithm 7 BiBitN with On-Line Summarization

Input:
$\mathcal{B}$: $m \times n$ binary matrix; m edges $\times$ n graphs
mnr: minimum number of rows
mnc: minimum number of columns
r: noise ratio
s: edgeset similarity threshold
Output:
$\mathcal{X}$: set of representative frequent edgesets

```
1.    for every edge pair (i, j) do
2.      S(i, j) = S(i) ∩ S(j)
3.      if S(i, j) is new and |S(i, j)| ≥ mnc then
4.        ⟨I, S(i, j)⟩ = ⟨{i, j}, S(i, j)⟩
5.        for every remainder edge, q ∈ E \ I   do
6.          if   |S(q) ∩ S(i, j)|/|S(i, j)| ≥ 1 − r then
7.              I = I ∪ q
8.          end if
9.        end for
10.       if |I| ≥ mnr and sim(I, I′) < s for every edgeset I′ in X then
11.         X = X ∪ ⟨I, S(i, j)⟩
12.       end if
13.     end if
14.   end for
```

6.3. Experimental Results

To evaluate the effect of the data summarization, we mined approximate frequent edgesets for various similarity threshold values. Moreover, to evaluate the overall effectiveness of the proposed approach, we mined approximate frequent dense modules from the 35 gene coexpression network described in section 4.3.1. We also conducted Gene Ontology enrichment analysis to assess the biological significance of the reported modules.

6.3.1. Effect of Edgeset Similarity Threshold

To see the effect of the data summarization, we ran the on-line summarization algorithm on the binary edge occurrence matrix constructed from the 35 gene coexpression networks, for support threshold values from 16 to 20, noise threshold 0.1, and edgeset similarity threshold values 0.5, 0.6, 0.7, and 0.8. Figure 6.1 shows hows the number of frequent edgesets varies for different edgeset similarity threshold values. We can see that the number of frequent edgesets decreases with increasing support threshold and increases with increasing edgeset similarity threshold. This is expected because less number of representative edgesets is needed for lower similarity threshold.

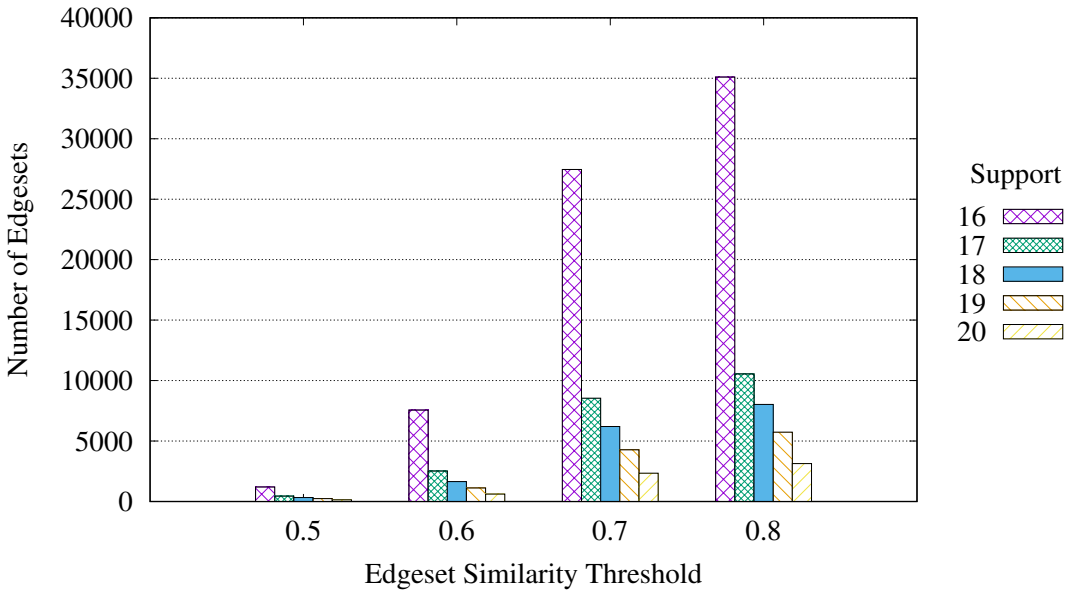


Figure 6.1. Number of frequent edgesets for varying edgeset similarity thresholds for biclustering with on-line summarization approach.

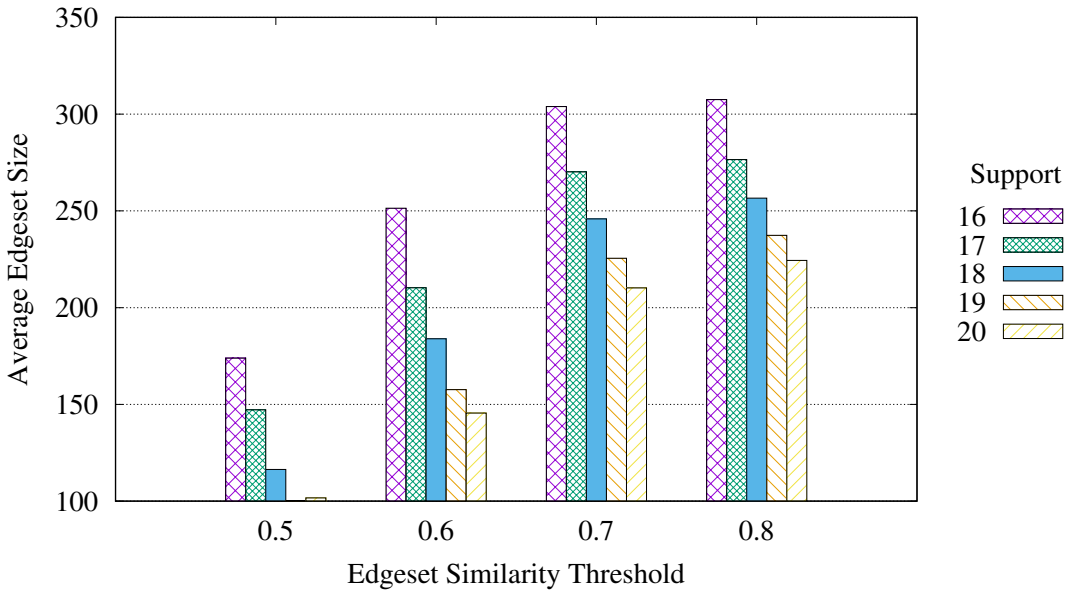


Figure 6.2. Average edgeset size for varying edgeset similarity thresholds for biclustering with on-line summarization approach.

Figure 6.2 shows how the average edgeset size varies for different edgeset similarity threshold values. We see that the average edgeset size increases with increasing edgeset similarity threshold.

Table 6.1. Comparison of the number of edgesets for support 20 for varying similarity thresholds

noise	0	0.1	0.2	0.3
Without summarization	3,004	3,153	3,224	3,244
With summarization ($s = 0.3$)	17	13	14	16
With summarization ($s = 0.4$)	61	38	31	38
With summarization ($s = 0.5$)	215	141	127	145
With summarization ($s = 0.6$)	579	613	599	826
With summarization ($s = 0.7$)	1,546	2,341	2,569	2,993
With summarization ($s = 0.8$)	2,789	3,138	3,221	3,244

To evaluate the effect of online frequent edgeset summarization, we mined approximate frequent edgesets and representative frequent edgesets for $minsup = 20$ We used edgeset similarity thresholds 0.3 to 0.8 for mining representative approximate frequent edgesets. Tables 6.1 shows the reported number of frequent edgesets for various similarity thresholds. The number of representative frequent edgeset increases as we increase the similarity thresholds. For a small similarity threshold, a small number of edgesets can claim to represent the entire set of approximate frequent edgesets. And for a large similarity threshold, fewer edgesets are similar to each other and thus the number of representative patterns is larger.

6.3.2. Topological Analysis of Frequent Edgesets

We ran the on-line summarization algorithm on the binary edge occurrence matrix for support threshold values from 16 to 20, noise threshold values from 0 to 0.3, and edgeset similarity threshold 0.6. Table 6.2 shows the topological properties of the frequent edgesets for varying parameter values. M is the number of frequent edgesets and $\overline{E}$ is the average number of edges in each frequent edgeset. Figure 6.3 shows how the number of frequent edgesets varies for different noise threshold values. There is a more noticeable increasing trend of the number of edgesets with increasing noise threshold, as compared with the BiBitN approach. Figure 6.4 shows how the average edgeset size varies for different noise threshold values. We see that the average edgeset size increases with increasing noise threshold, as expected.

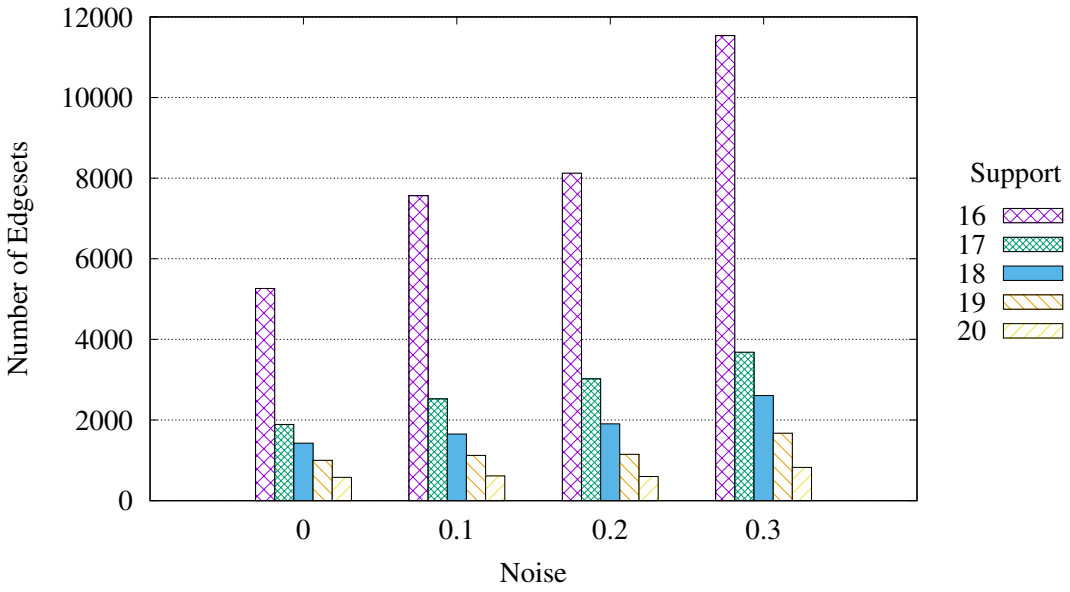


Figure 6.3. Number of frequent edgesets for varying noise thresholds for biclustering with on-line summarization approach with edgeset similarity threshold 0.6.

We can see that compared to the BiBitN approach, this approach produces much smaller number of frequent edgesets, especially for low values of edgeset similarity threshold. This makes analysis with lower values of support threshold more feasible.

Table 6.2. Topological properties of the frequent edgesets for biclustering with on-line summarization approach with edgeset similarity threshold 0.6. M is the number of frequent edgesets and $\overline{E}$ is the average number of edges in each frequent edgeset.

noise	0		0.1		0.2		0.3	
minsup	M	$\overline{E}$	M	$\overline{E}$	M	$\overline{E}$	M	$\overline{E}$
16	5.3 K	55.8	7.6 K	251.4	8.1 K	446.9	11.5 K	1.2 K
17	1.9 K	51.6	2.5 K	210.3	3 K	442.9	3.7 K	982.8
18	1.4 K	47	1.7 K	183.9	1.9 K	482.6	2.6 K	928.1
19	999	43.8	1.1 K	157.6	1.1 K	401.3	1.7 K	967.7
20	579	43.7	613	145.5	599	349.8	826	848.1

6.3.3. Topological Analysis of Frequent Dense Modules

We mined dense modules from the subgraphs induced by the frequent edgesets, using density thresholds 0.5 and 0.6, and only modules of size four or larger were considered. Table 6.3 shows

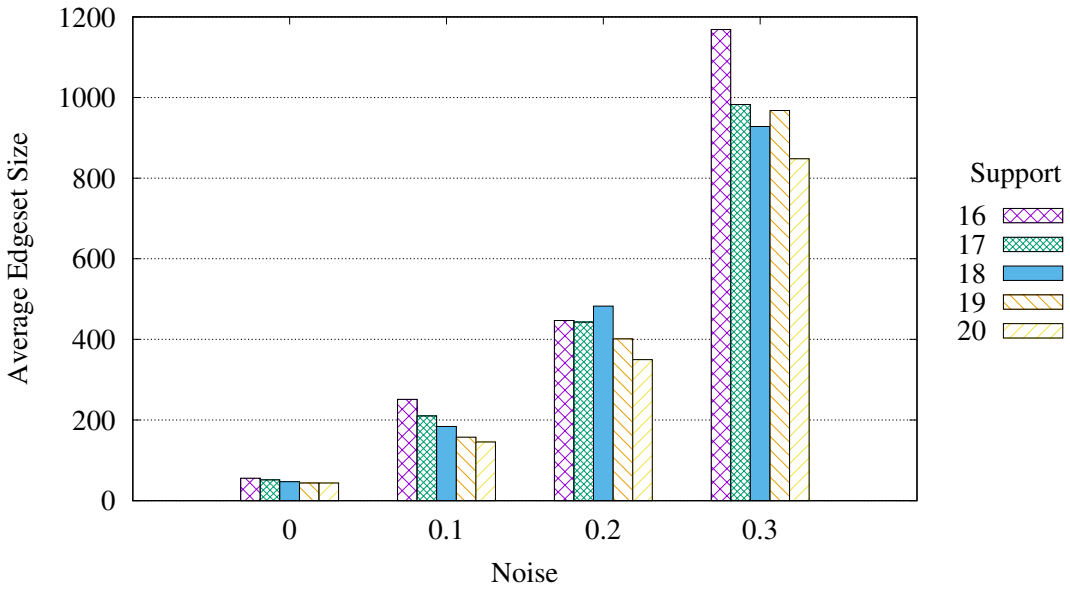


Figure 6.4. Average edgeset size for varying noise thresholds for biclustering with on-line summarization approach with edgeset similarity threshold 0.6.

the topological properties of the frequent dense modules for varying parameter values and edgeset similarity threshold 0.6. M' denotes the number of approximate frequent edgesets that have at least one dense module for the specified density threshold, $\overline{DM}$ denotes the average number of dense modules in edge-induced subgraph of each edgeset, and $\overline{V'}$ denotes the average size of the dense modules. It shows the same general trend as the BiBitN results. That is, the number of edgesets with at least one dense module and the average number of dense modules both decrease as the support threshold is increased.

Table 6.4 shows the number of all frequent dense modules, unique frequent dense modules, and the percentage of the unique frequent dense modules. As with the BiBitN approach, the percentage of the unique dense modules increases with increasing support threshold. We can see that the percentages of the unique dense modules are much higher compared to the BiBitN results, which means there are much smaller number of duplicate modules.

6.3.4. Gene Ontology Enrichment Analysis

To assess the biological significance of the reported modules, we conducted Gene Ontology enrichment analysis the same way as described in section 4.3.4. Table 6.5 shows the percentage of frequent dense modules that are biologically enriched. E_{MF} and E_{KEGG} denote the percent enriched in molecular functions and KEGG pathways respectively. As with the BiBitN results,

Table 6.3. Topological properties of the frequent dense modules for biclustering with on-line summarization approach with edgeset similarity threshold 0.6. M' is the number of frequent edgesets that have at least one dense module, $\overline{DM}$ is the average number of dense modules in each edge-induced subgraph, and $\overline{V'}$ is the average size of the dense modules.

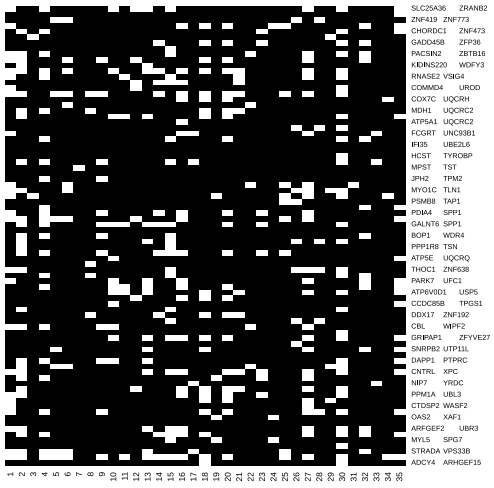
noise		0			0.1			0.2		
minsup	density	M'	$\overline{DM}$	$\overline{V'}$	M'	$\overline{DM}$	$\overline{V'}$	M'	$\overline{DM}$	$\overline{V'}$
17	0.5	513	11.2	4	2.3 K	51.8	4.2	2.9 K	206.7	4.3
	0.6	20	1.2	4	646	4.7	4.2	1.7 K	15.2	4.4
18	0.5	346	10.6	4	1.4 K	42.8	4.1	1.8 K	252	4.3
	0.6	6	1.2	4.2	362	4	4.2	1.1 K	18.9	4.4
19	0.5	238	9.3	4	941	32.7	4.1	1.1 K	187.8	4.3
	0.6	6	1	4	190	3.2	4.2	579	14.6	4.4
20	0.5	134	9.9	4	499	29.5	4.1	540	153.6	4.2
	0.6	3	1	4	84	3.3	4.2	265	13.1	4.3

modules with smaller noise ratio have higher enrichment, and the enrichment with GO molecular functions is higher than the enrichment with KEGG pathways. Table 6.6 shows the top biological signatures that were enriched the most in the reported modules for $sup = 17$, $noise = 0.1$, and $density = 0.5$.

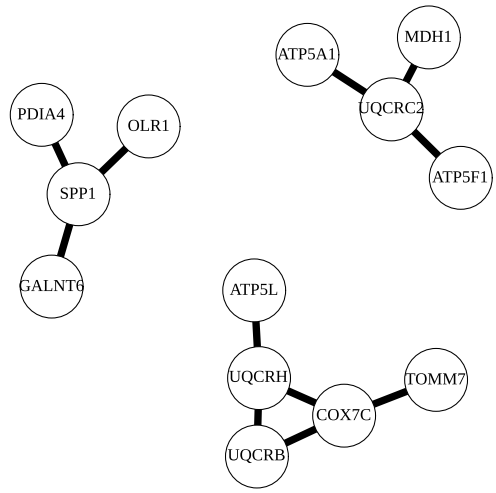
Figure 6.5 shows an example of an approximate frequent edgeset for $sup = 17$, $noise = 0.1$. (a) show the submatrix of the edge occurrence matrix that shows the occurrences of edges of the edgeset in the 35 networks. The rows correspond to the edges in the edgeset, and the columns correspond to coexpression networks. (b) shows the dense modules mined from the subgraph induced by the edgeset, using density 0.5. Nodes are labeled by their corresponding gene identifiers. The genes in this representative approximate edgeset are enriched with five KEGG pathways; Oxidative Phosphorylation, Cardiac Muscle Contraction, Alzheimers Disease, Parkinsons Disease, and Huntingtons Disease, and two molecular functions; Electron Transfer Activity, and Oxidoreductase Activity.

Table 6.4. Number of all frequent dense modules, unique frequent dense modules, and the percentage of the unique frequent dense modules for biclustering with on-line summarization approach with similarity threshold 0.6.

noise		0			0.1			0.2		
minsup	density	all	unique	%	all	unique	%	all	unique	%
17	0.5	5.8 K	434	7.5	117.5 K	5.7 K	4.8	591.1 K	26.6 K	4.5
	0.6	25	11	44	3 K	343	11.4	25.4 K	2 K	8
18	0.5	3.7 K	305	8.3	61.3 K	3.5 K	5.7	449.5 K	25.7 K	5.7
	0.6	7	6	85.7	1.4 K	217	15.1	20 K	1.9 K	9.7
19	0.5	2.2 K	203	9.2	30.8 K	2 K	6.6	198 K	15.3 K	7.7
	0.6	6	4	66.7	599	111	18.5	8.4 K	1.1 K	13.3
20	0.5	1.3 K	155	11.6	14.7 K	1.3 K	9	83 K	8.4 K	10.2
	0.6	3	3	100	274	57	20.8	3.5 K	602	17.3



(a) Submatrix for Frequent Edgeset



(b) Frequent Dense Modules

Figure 6.5. Sample frequent edgeset for $minsup = 17$ and $noise = 0.1$, and dense modules in the edgeset for $density = 0.5$

Table 6.5. GO term enrichment analysis for frequent dense modules for biclustering with online summarization approach with edgeset similarity threshold 0.6. E_{MF} and E_{KEGG} denote the percent enriched in molecular functions and KEGG pathways respectively.

noise		0		0.1		0.2	
minsup	density	E_{MF}	E_{KEGG}	E_{MF}	E_{KEGG}	E_{MF}	E_{KEGG}
17	0.5	80.4	64.3	65.5	55.7	51	43.4
	0.6	90.9	45.5	80.5	60.3	68.9	49.8
18	0.5	81.6	62.6	71.1	59.6	51.3	43
	0.6	100	50	83.9	56.7	68.5	49.5
19	0.5	87.2	66.5	75	61.9	55.2	46.7
	0.6	100	50	83.8	54.1	71.7	53.3
20	0.5	85.8	67.1	77.5	66.5	61.6	52
	0.6	100	33.3	91.2	66.7	76.4	56.1

Table 6.6. Top enriched biological signatures in the reported modules for $sup = 17$, $noise = 0.1$, and $density = 0.5$ for biclustering with on-line summarization approach with edgeset similarity threshold 0.6.

GO Molecular Function	Count
STRUCTURAL CONSTITUENT OF RIBOSOME	1996
RRNA BINDING	503
5S RRNA BINDING	277
ELECTRON TRANSFER ACTIVITY	271
OXIDOREDUCTASE ACTIVITY ACTING ON NAD P H	214
NADH DEHYDROGENASE ACTIVITY	208
ANTIGEN BINDING	194
IMMUNOGLOBULIN RECEPTOR BINDING	175
KEGG pathway	**Count**
RIBOSOME	2001
HUNTINGTONS DISEASE	503
OXIDATIVE PHOSPHORYLATION	493
PARKINSONS DISEASE	472
ALZHEIMERS DISEASE	464
CARDIAC MUSCLE CONTRACTION	240
AUTOIMMUNE THYROID DISEASE	58
MAPK SIGNALING PATHWAY	54
AMINOACYL TRNA BIOSYNTHESIS	52
PROTEIN EXPORT	41

7. CONCLUSION AND FUTURE WORK

Mining frequent dense modules from multiple gene coexpression networks has applications in functional gene annotation and biomarker discovery. In this thesis, we proposed biclustering-based approaches for mining such modules. In the first algorithm, we first mine biclusters with high density of ones, which correspond to approximate frequent edgesets, and then extract dense modules from these edgesets. In this approach, a large number of frequent edgesets have high overlap, resulting in many duplicate modules in the final result. Thus, we proposed two algorithms that extend the first algorithm and use post-processing and on-line data summarization methods to mine set of representative frequent edgesets. These extended algorithms mine reduced number of duplicate modules, which makes analyses feasible for lower support thresholds. Since the proposed approaches only explore a small part of the frequent patterns search space, the running times are extremely fast. Experiments on real gene coexpression networks show that the reported frequent dense modules are biologically enriched with known KEGG pathways and molecular functions.

Future work can include incorporating the edge connectivity in biclustering process. In the proposed algorithms, we mine edgesets by only considering edge occurrences in relation graph set, and not the edge connectivity. If we can efficiently incorporate the edge connectivity in the biclustering process, we can mine connected edgesets. This will likely speed up the biclustering run time, since only the edges connected to the patterns need to be considered. It may also reduce the size of the edgesets, making analyses easier. Moreover, if we can incorporate density constraint in the process, we may be able to skip the dense module mining step.

REFERENCES

[1] Gary D. Bader and Christopher W.V. Hogu. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(2), 2003.

[2] Alvis Brazma and Jaak Vilo. Gene expression data analysis. *FEBS Letters*, 480(1):17–24, 2000.

[3] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[5] 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.

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

[7] Logan Everett, Li-San Wang, and Sridhar Hannenhalli. Dense subgraph computation via stochastic search: Application to detect transcriptional modules. *Bioinformatics (Oxford, England)*, 22:e117–23, 08 2006.

[8] Elisabeth Georgii, Sabine Dietmann, Takeaki Uno, Philipp Pagel, and Koji Tsuda. Enumeration of condition-dependent dense modules in protein interaction networks. *Bioinformatics*, 25(7):933–940, 2009.

[9] B. Goethals. Survey on frequent pattern mining. *University of Helsinki*, 19:840–852, 2003.

[10] Aditya Goparaju, Tyler Brazier, and Saeed Salem. Mining representative maximal dense cohesive subnetworks. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 4(1):1–11, 2015.

[11] Karam Gouda and Mohammed J. Zaki. GenMax: An efficient algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery: An International Journal*, 11(3):223–242, Nov 2005.

[12] Laurie J. Heyer, Semyon Kruglyak, and Shibu Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome research*, 9 11:1106–15, 1999.

[13] Haiyan Hu, Xifeng Yan, Yu Huang, and Xianghong Jasmine Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21 Suppl 1:i213–i221, 2005.

[14] 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–i229, 2007.

[15] Daxin Jiang, Jian Pei, and Aidong Zhang. Dhc: A density-based hierarchical clustering method for time series gene expression data. In *Proceedings of the 3rd IEEE Symposium on BioInformatics and BioEngineering*, BIBE '03, pages 393–, Washington, DC, USA, 2003. IEEE Computer Society.

[16] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 16(11):1370–1386, November 2004.

[17] Yuval Kluger, Ronen Basri, Joseph Chang, and Mark Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome research*, 13:703–16, 05 2003.

[18] Mehmet Koyuturk, Ananth Grama, and Wojciech Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *Bioinformatics*, 20(Suppl 1):i200–i207, 2004.

[19] Mehmet Koyutürk, Wojciech Szpankowski, and Ananth Grama. Biclustering gene-feature matrices for statistically significant dense patterns. pages 480 – 484, 09 2004.

[20] Homin K. Lee, Amy K. Hsu, Jon Sajdak, Jie Qin, and Paul Pavlidis. Coexpression analysis of human genes across many microarray data sets. *Genome Res.*, 14(6):1085–1094, 2004.

[21] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[22] Oliver Mason and Mark Verwoerd. Graph theory and networks in biology. *IET systems biology*, 1:89–119, 04 2007.

[23] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[24] Emma Pierson, the GTEx Consortium, Daphne Koller, Alexis Battle, and Sara Mostafavi. Sharing and specificity of co-expression networks across 35 human tissues. *PLOS Computational Biology*, 11(5):1–19, 05 2015.

[25] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 02 2006.

[26] Domingo S. Rodriguez-Baena, Antonio J. Perez-Pulido, and Jesus S. Aguilar-Ruiz. A biclustering algorithm for extracting bit-patterns from binary datasets. *Bioinformatics*, 27(19):2738–2745, October 2011.

[27] Saeed Salem. Template edge similarity graph clustering for mining multiple gene expression datasets. *International Journal of Data Mining and Bioinformatics*, 18(1):28–39, 2017.

[28] Saeed Salem and Cagri Ozcaglar. MFMS: Maximal frequent module set mining from multiple human gene expression data sets. In *Proceedings of the 12th International Workshop on Data Mining in Bioinformatics*, BioKDD '13, pages 51–57, New York, NY, USA, 2013. ACM.

[29] Saeed Salem and Cagri Ozcaglar. Hybrid coexpression link similarity graph clustering for mining biological modules from multiple gene expression datasets. *BioData Mining*, 7(1):16, 2014.

[30] Roded Sharan and Ron Shamir. Center click: A clustering algorithm with applications to gene expression analysis. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 307–316. AAAI Press, 2000.

[31] Heather L. Turner, Trevor C. Bailey, and Wojtek J. Krzanowski. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational Statistics Data Analysis*, 48:235–254, 2005.

[32] Miranda Uitert, Wouter Meuleman, and Lodewyk Wessels. Biclustering sparse binary genomic data. *Journal of computational biology : a journal of computational molecular cell biology*, 15:1329–45, 01 2009.

[33] Xifeng Yan, Xianghong Jasmine Zhou, and Jiawei Han. Mining closed relational graphs with connectivity constraints. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 324–333, New York, NY, USA, 2005. ACM.

[34] Bin Zhang and Steve Horvath. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4, 2005.