CONTEXT SPECIFIC MODULE MINING FROM MULTIPLE CO-EXPRESSION GRAPH

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

By

Md Shakhawat Hossain

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

Major Department:
Computer Science

May 2017

Fargo, North Dakota

NORTH DAKOTA STATE UNIVERSITY

Graduate School

Title

CONTEXT SPECIFIC MODULE MINING FROM MULTIPLE CO-EXPRESSION GRAPH

By

Md Shakhawat Hossain

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:

Prof. Saeed Salem
Chair

Prof. Simone Ludwig

Prof. Mukhlesur Rahman

Approved:

August 14, 2017
Date

Prof. Kendall Nygard
Department Chair

ABSTRACT

Gene co-expression networks can be used to associate genes of unknown function with biological processes or to find genes in a specific context, environment responsible for a disease. We provide an overview of methods and tools used to identify such recurrent patterns across multiple networks,can be used to discover biological modules in co-expression networks constructed from gene expression data and we explain how this can be used to identify genes with a regulatory role in disease. However, existing algorithms are very much costly in terms of time and space. As network size or number increases, mining such modules get much more complex. We have developed an efficient approach to mine such recurrent context specific modules from 35 gene networks.This computationally very difficult problem due to the exponential number of patterns was solved non-exponentially.

ACKNOWLEDGEMENTS

I would like to be highly thankful to my supervisor, Dr. Saeed Salem. His guidance was invaluable in helping me finish this paper. His enthusiasm in and profound knowledge of the discipline of computer science are a source of inspiration to me.

I am very grateful to Dr. Kendall Nygard . Without him, I would not have been introduced to this great university. His unfailing support and advising enabled me to successfully overcome the challenges throughout my graduate studies.

I would like to express my gratitude to my committee members, Dr. Simone Ludwig and Dr. Mukhlesur Rahman, for taking the time to help me graduate from the University.I also want to thank North Dakota State University and the Department of Computer Science for offering me the opportunity to study at such a great school, full of smart and diverse people who contributed to a great environment for research and study.

DEDICATION

To Almighty God,mother and father, for supporting me all the way.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION AND RELATED WORK

Gene co-expression networks are usually constructed using data sets generated by high-throughput gene expression profiling technologies such as Micro array or RNA-Seq. A gene co-expression network (GCN) is an undirected graph, where each node corresponds to a gene, and a pair of nodes is connected with an edge if there is a significant co-expression relationship between them.Large protein networks have only recently become available for human [11],enabling new opportunities for elucidating pathways involved in major diseases and pathologies [17]. Studying the building principles of biological networks could potentially revolutionize our view of biology and disease pathologies[3]. Due to the noisy nature of high throughput data, a significant number of spurious edges exist in biological networks, which may lead to the discovery of false patterns if we just extract modules from a single network. Since biological modules are expected to be active across multiple conditions, we can easily filter out spurious interaction between genes -which is represented by an edge- by mining frequent patterns in multiple biological networks simultaneously. Based on the recurrent network patterns, we performed functional annotation, this approach can address the problems above:(i)to separate true functional links from spurious co-expression links. We suggest that a co-expression link recurrent in multiple data sets is more likely to represent a true functional link. (ii) To identify functionally related genes without direct co-expression, when we combine multiple expression networks , subtle signals may emerge that cannot be identified in any of the individual networks. Such signals include recurrent paths that may extend beyond simple co-expression clusters yet represent functional modules. (iii) To conditionally annotate gene functions. Because a gene cannot exert its function by itself but instead does so by interaction with other genes, its functional switch is likely to be caused by or result from the alteration of its interaction partners[23].As a solution to mining dense module from multiple biological network could be aggregating these networks together and identify dense subgraphs in the aggregated graph. However, it could result in false dense subgraphs that may not occur frequently in the original networks[22]. Figure 1.1a illustrates such an example with a cartoon of six graphs. If we simply add these graphs together to construct a summary graph, we may find a dense subgraph comprising vertices a, b, c, and d. Unfortunately, this subgraph is neither dense nor frequent in the original graphs.

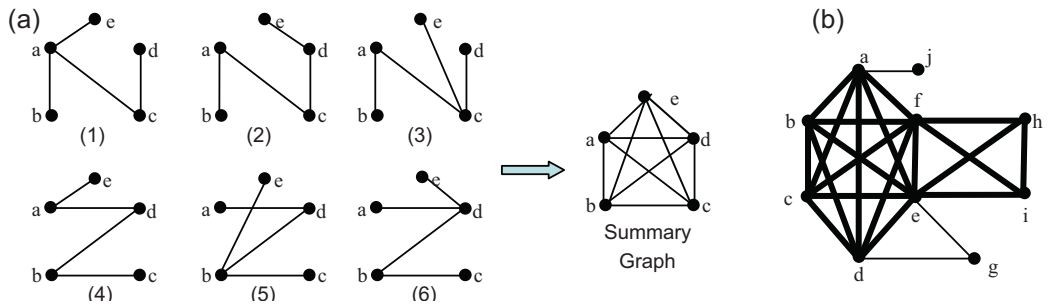


Figure 1.1. Given six graphs with the same vertex set but different edge sets, we construct a summary graph by adding these six graphs together.The dense subgraph in the summary graph $\{a, b, c, d\}$ actually does not occur in any original graph. (b) The vertices e and f are shared by cliques $\{a, b, c, d, e, f\}$ and $\{e, f, h, i\}$; they can be assigned to both cliques only by approaches that are able to detect overlapping dense subgraphs (cliques are the densest subgraphs)

A potential solution to the false pattern problem could be mining frequent subgraphs directly. A subgraph is frequent if it occurs multiple times in a set of graphs. Frequent subgraph discovery in general is considered a hard problem. However, biological networks can often be modeled as a special class of graph where each gene occurs once and only once in a graph. That means, our graph has distinct node labels and we do not have the "subgraph isomorphism problem" which is NP-hard and so far constitutes the bottleneck of subgraph frequency counting.Recently, we and others have designed efficient approaches to identify frequent subgraphs across multiple relation networks by decomposing the networks into smaller pieces and applying pattern expansion techniques [13] [21], or by performing frequent set mining with subsequent connectivity checking [9]. However, these approaches encounter scalability and interpretability issues when being applied to massive biological networks: (1) In both approaches when tested, the time and memory requirements increase exponentially with increasing size of patterns and increasing number of networks. The number of frequent dense subgraphs is explosive when there are very large frequent dense subgraphs, e.g., subgraphs with hundreds of edges. (2) A frequent dense subgraph may not represent a tight association among its nodes.

A dense sub graph is a sub graph that satisfies a user defined constraint [5]. In a PPI (Protein–protein interaction) network, proteins which are members of the same dense sub graph can represent protein complexes [15]. Although, there are ways of detecting protein complexes experimentally, there are weaknesses to each of those ways [10]. Thus it is important to use data currently available and extrapolate other proteins that might belong to these complexes and find

new complexes. This can help identify disease causing genes [20] to identify people at risk and start preventative treatment of possible future conditions.

In this paper, we addressed the aforementioned two issues and developed a novel algorithm, called "MineFrequent", to implement this idea. We developed a data mining procedure based on frequent itemset mining (FIM) and hill climbing (greedy local search) approach to extensively discover network patterns that recur in at least certain number of graphs. We attempted to represent each edge as a binary matrix, where each column represents occurrence of all edges in a particular graph and from that matrix we efficiently extracted 3 connected nodes (genes) or two connected edges with a common gene and that root appear in at least certain number of graph (frequency threshold) . From this edge-graph matrix, using each three node seed, we followed a greedy hill climbing approach based on a predefined density threshold and extracted dense subgraphs.

2. FREQUENT ITEMSET MINING

Starting from a single node and then enumerate all other possible nodes with the existing nodes of a graph and looking them inside at least certain number of graph can be mapped to itemset mining problem . This collection of all possible combinations of items of a set S is called the *power set*, or $P(S)$, of S [18]. A set enumeration tree is an efficient way to enumerate the *power set* of a set. Creating a set enumeration tree is efficient and makes sure no combination is enumerated twice [12]. An example of a set enumeration tree for the set {A,B,C,D} is shown in Figure 2.1.

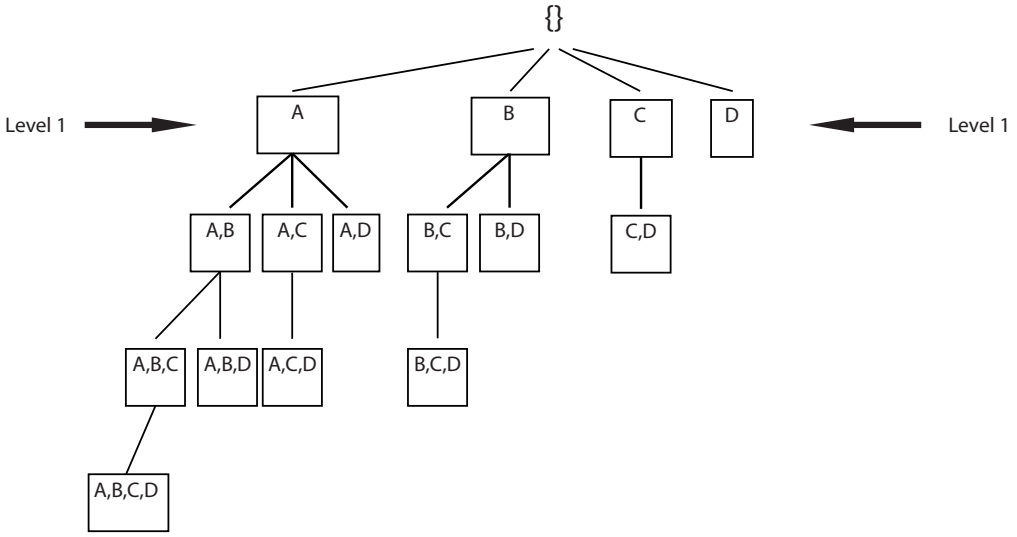


Figure 2.1. Set enumeration tree for {A,B,C,D}

To create a set enumeration tree, the set is first sorted and the root of the tree is set as *null*. Then the first level nodes of tree is created. Each node in the first level contains a different item of the set in a sorted order; this set is called the *member set*. The tree can be traversed in a breadth-first or a depth-first approach. In Figure 2.1, the depth-first traversal order would be for the first level node with A as the member set would be {A}, {A,B}, {A,B,C}, {A,B,C,D}, {A,B,D}, {A,C}, {A,C,D}, {A,D}. Then the tree would go to node {B} on the first level.

[1] showed how to use the set enumeration tree for finding *frequent itemsets* in association rule mining. An example of frequent itemset mining is shown in Figure 2.2. A number of *transactions* containing items are mined. One of the parameters of frequent itemset mining is the minimum

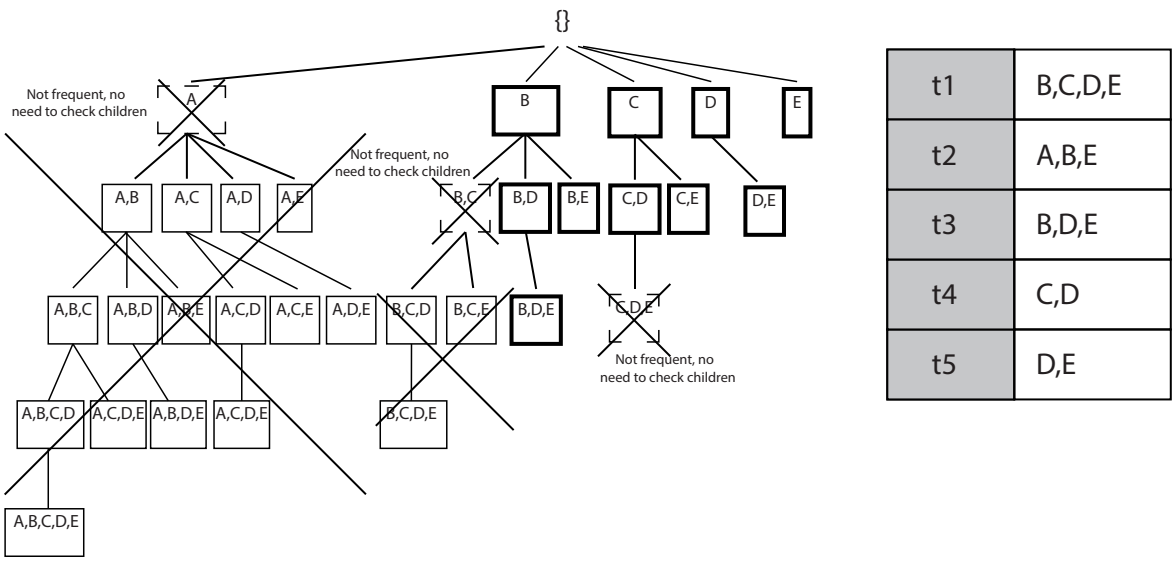


Figure 2.2. Frequent itemset enumeration tree with minimum support of 2

support. An itemset is frequent if the number of transactions in which the itemset appears is greater than or equal to the minimum support. Using the set enumeration tree, the member set of each node is tested for frequency. In Figure 2.1, the search space is 15 nodes with a set size of 4; in Figure 2.2, the search space is 30 nodes with a set size of 5. With the search space growing vastly with each increase in set size, there is a need to prune the search space to avoid taking too much time. In frequent itemset mining, the frequency property is *anti-monotone*, thus if an itemset is not frequent, no superset of that itemset will be frequent as well. In the itemset enumeration tree, this allows for pruning of all children of a node if the node is not frequent. In Figure 2.2, the search branches rooted at {A}, {B,C}, and {C,D,E} are pruned this way. The frequent itemsets are shown as bolded boxes in Figure 2.2.

[Anti-monotone constraint] A constraint P is anti-monotone for an itemset, $V \subseteq \mathcal{V}$, if the following condition is satisfied:

$$P(V) = TRUE \Rightarrow P(V') = TRUE, \forall_{V'} : V' \subseteq V$$

We can see that the frequency constraint is anti-monotone and that is why we can employ it in pruning search branches. Identifying small groups of related members are not very useful; instead the largest groups that still satisfy the frequent property are more interesting. Thus the concept of *maximal* frequent itemsets is introduced. An itemset is maximally frequent if there exists no superset of that itemset that is frequent. Figure 2.3 shows the maximal itemsets as bolded boxes.

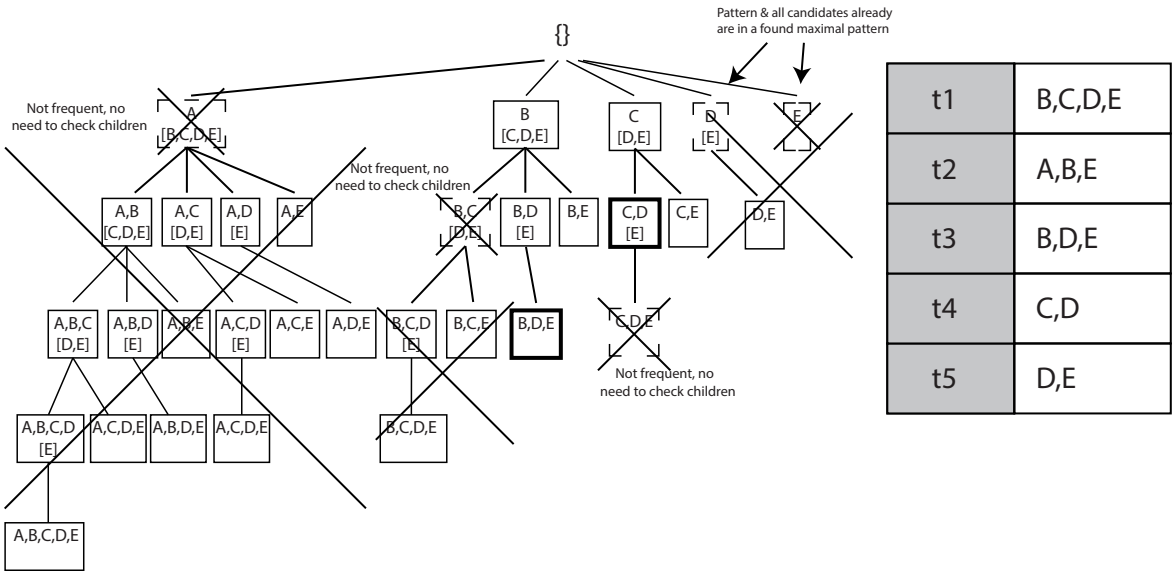


Figure 2.3. Maximal frequent itemset enumeration tree with minimum support of 2.

[Maximal frequent] An itemset, $S \subseteq \mathcal{S}$, is maximal if the following condition is satisfied:

$$FREQ(S) = TRUE, \nexists S' \supseteq S \wedge FREQ(S') = TRUE$$

Enumerating only maximal itemsets offers a few more opportunities for pruning. Any node with a frequent child cannot be maximal. After pruning for frequency, only the leaf nodes are potential maximal frequent nodes. Also, if a node's member set combined with all possible extensions is a subset of a discovered maximal set, then the node and its children can be pruned; as the node and its children's member sets will be subsets of that maximal set.

2.1. Hill-climbing (Greedy Local Search)

In Computer Science , hill climbing is a mathematical optimization technique which belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found. Hill climbing achieves optimal solutions in convex problems – otherwise it will find only local optima (solutions that cannot be improved by considering a neighbouring configuration), which are not necessarily the best possible solution (the global optimum) out of all possible solutions (the search space) To attempt overcoming being stuck in local optima, one could use restarts (i.e. repeated local search), or more complex schemes

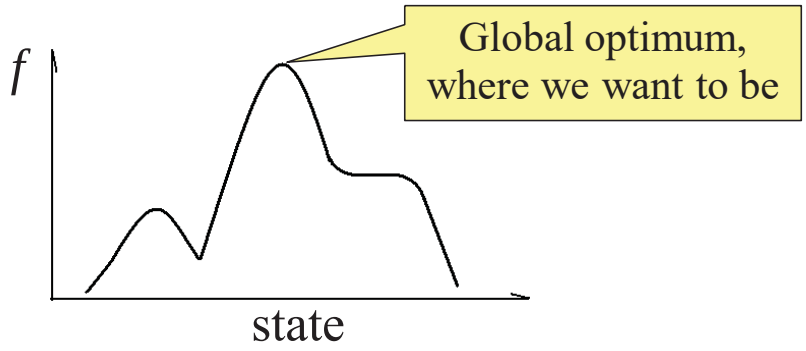


Figure 2.4. Hill climbing search

based on iterations (like iterated local search), or on memory (like reactive search optimization and tabu search), or on memory-less stochastic modifications (like simulated annealing) The relative simplicity of the algorithm makes it a popular first choice amongst optimizing algorithms. It is used widely in artificial intelligence, for reaching a goal state from a starting node.

This strategy is also widely applied for minimizing problem.

2.2. Related Work

In a paper published at 2009 Xianghong Jasmine Zhou [6] explained that to solve the issue of subgraph discovery across multiple graph network could be mining frequent subgraphs directly. In this way, the issue of false pattern problem or false interaction among genes can easily be overcome and we could get the true interaction between certain group of genes contributing to a certain disease or certain phenotype. A subgraph is frequent if it occurs multiple times in a set of graphs. Frequent subgraph discovery in general is considered a hard problem. However, biological networks can often be modeled as a special class of graph where each gene occurs once and only once in a graph. That means, all their and our graphs(co -expression networks) has distinct node labels, and we do not have the "subgraph isomorphism problem" which is NP-hard and so far constitutes the bottleneck of subgraph frequency counting.

Recently, Xianghong Jasmine Zhou [6] and others have designed efficient approaches to identify frequent subgraphs across multiple relation networks by decomposing the networks into smaller pieces and applying pattern expansion techniques [13] [21], or by performing frequent set mining with subsequent connectivity checking [9]. In all the mentioned approaches, time and memory requirements increase exponentially with increasing size of patterns and increasing number of

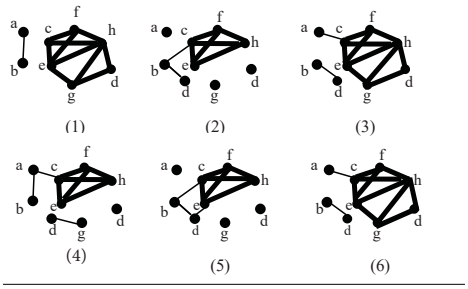


Figure 2.5. Shown are six graphs with the same vertex set but different edge sets. The bold subgraph $\{c-e, c-f, c-h, f-e, f-h, e-h, e-g, g-h, h-d, g-d\}$ occurs in three out of the six graphs (graphs 1,3, and 6). However, the vertices/edges of this subgraph may not be tightly associated in their occurrence, because one large component, the subgraph $\{c-e, c-f, c-h, f-e, f-h, e-h\}$ occurs in every network

networks.

In the same paper Xianghong Jasmine Zhou [6], they also mentioned that, a frequent(if we consider a certain frequency threshold as a definition of frequent) dense subgraph may not represent a tight association among its nodes. Figure 2.5 shows a sample network dataset. Vertices e, c, f, h, d, and g form a frequent dense subgraph. However,biologically it is more interesting to divide this subgraph into two modules, one comprising e, c, f, and h; the other comprising h, d, g, and e since these two modules have different **occurrences** throughout this graph set (details see the figure caption). As one can see, frequent dense subgraphs may not capture accurate information for the discovery of biological modules.

The bottleneck of subgraph frequency counting represented mining **coherent** dense subgraphs, a concept having better **interpretability** than frequent graph. All edges in a **coherent** subgraph should exhibit correlated occurrences in the whole graph set. We also term this kind of subgraph **"Network Module"**. This approach is known as CODENSE [6]. According to the definition of **coherent** dense subgraph, we are able to distinguish the two modules shown in Figure 2.5. Moreover, the design of CODENSE can solve the scalability issue. Instead of mining each biological network individually, CODENSE compresses the networks into two meta-graphs and performs clustering in these two graphs only. Thus, CODENSE can handle any large number of networks. Using CODENSE, we can successfully identify high-quality network modules within limited time and memory.

As a side product, CODENSE also provides a solution to a graph mining problem-discovery

of overlapping graph clusters. It is known that under different conditions, one gene may serve different roles and be involved in different functional groups [4]; thus identifying overlapping clusters is important in biological applications.

They developed a data mining procedure based on frequent itemset mining FIM and biclustering to extensively to discover network patterns that recur in at least five datasets and a biclustering algorithm based on simulated annealing to discover frequent edge sets. More precisely, they employed simulated annealing to maximize the objective function $\frac{c'}{mn+\lambda \cdot c}$, where c' is the number of 1s in the input matrix, c, m and n are the numbers of 1s, rows and columns of the bicluster, respectively, and λ is a regularization factor. Clearly, such an objective function is in favor of biclusters with a high density of 1 and with large size. Note that, the density is maximized to 1 when $c' = mn$, while the size of bicluster is maximized when $c' = c$ i.e. the pattern is as large as the input matrix [7].

The rest of the paper is structured as follows. Section 3 gives the problem definition and algorithm. The results of the algorithm on real world data is shown in Section 4. Lastly, Section 5 presents the conclusion and how this work might be extended in the future.

3. PROBLEM DESCRIPTION

In this section, I define some terms that are used throughout the paper. The graphs considered for this paper are *simple* graphs. A simple graph is a graph which only has undirected edges. In addition, a simple graph has no self-directed edges or multi-edges.

A graph $G = (V, E)$, contains a set of vertices $V = \{v_1, v_2, \cdots, v_m\}$, and a set of edges $E = \{e_1, e_2, \cdots, e_p\}$ connecting the vertices. Two vertices u and v are *adjacent* if there is an edge connecting u and v. The degree of a vertex v is denoted by $deg(v)$ and is the number of edges connected to v. A relation graph network set consists of n undirected simple graphs, $D = \{G_i = (V, E_i)\}$, i = 1,. . . ,n, where a common vertex set V is shared by the graphs in the set. We denote the vertex set of a graph G by $V(G)$ and the edge set by $E(G)$. Let $w_i(u, v)$ be the weight of an edge $e_i(u, v)$ in G_i. For an unweighted graph, $w_i(u, v)$ is equal to 1 if there is an edge between u and v, otherwise 0. We chose to illustrate the principles on unweighted and undirected graphs in this paper, although our algorithm should be extendable to weighted and directed graphs.

[γ- modules] A module of a graph is a generalization of a connected component. A connected component has the property that it is a set X of vertices such that every member of X is a non-neighbor of every vertex not in X. (It is a union of connected components if and only if it has this property.)More generally, X is a module if, for each vertex v not in X, either every member of X is a non-neighbor of v or every member of X is a neighbor of v. Equivalently, X is a module if all members of X have the same set of neighbors among vertices not in X. where γ represent the connectivity of edges in that module across the graph network.In Figure 3.2, some sample modules are shown from a sample graph network.

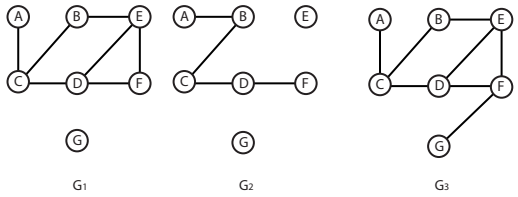


Figure 3.1. A gene co-expression network

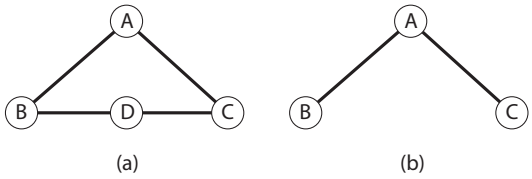


Figure 3.2. Example of modules

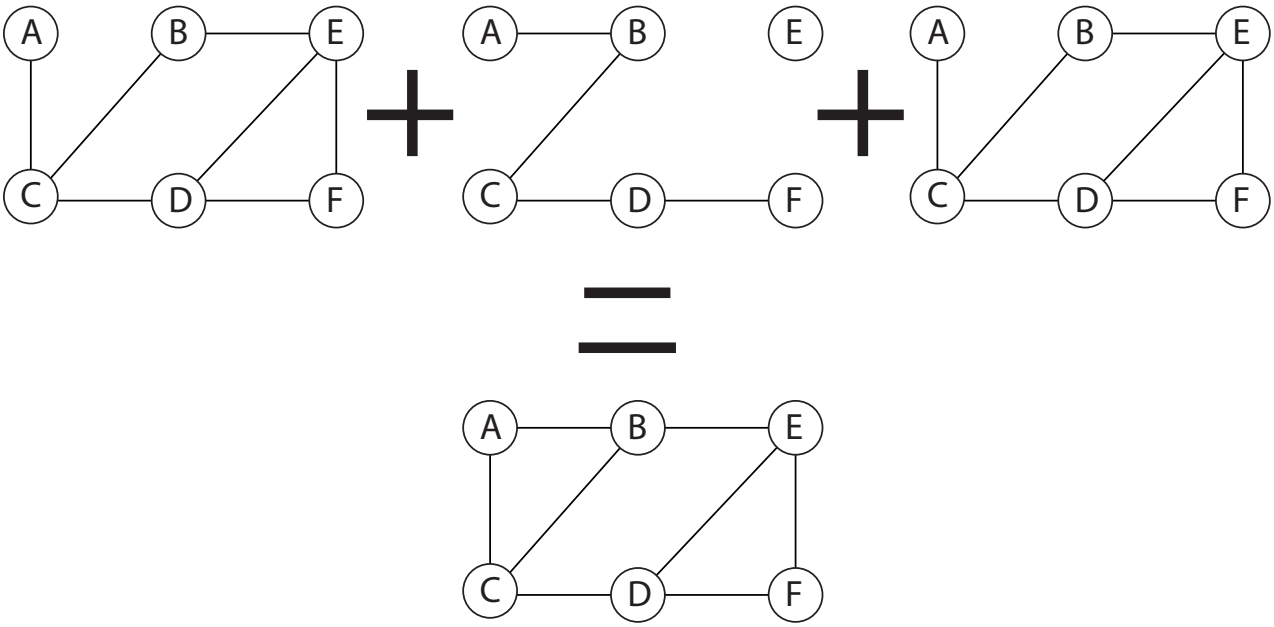


Figure 3.3. Summary graph.

[Summary Graph] Given a relation graph dataset, $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i = (V, E_i)$, the summary graph of D is an unweighted graph $\hat{G} = (V, \hat{E})$ where an edge e_i is present if it occurs in more than k graphs in D, where k is a user-defined support threshold.see an example in Figure 1.1a.

[Edge Occurence Matrix] An edge occurence matrix M_E,is a matrix $M_{e\times g}$,where $m_{i\times j}$ is a single entry in the matrix at it i^{th} row and j^{th} column , contains a binary value 0 or 1, where e represent the number of frequent edges in that dense graph under that frequency threshold and g represent number of graph in that co-expression network.In Figure 3.4 , we can see an edge occurrence matrix.

[Edge Support Vector] Given a relation graph dataset, $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i = (V, E_i)$, the support vector of an edge e, written w(e), is of length n where n is the number of graphs. The i-th element of w(e) corresponds to the weight of edge e in the i-th graph. The support vector of the edge (a, b) for the six graphs shown in Figure 1.1a is [1, 1, 1, 0, 0, 0], while the support vector of the edge (b, c) is [0, 0, 0, 1, 1, 1]. As one can see, edges (a, b) and (b, c) are not correlated in this dataset,though both of them are frequent. We used this edge support vector to calculate A_i as mentioned in Algorithm 2.

[Coherent Graph] Given a relation graph dataset, $D = \{G_1, G_2, \ldots, G_n\}$, where $G_i =$

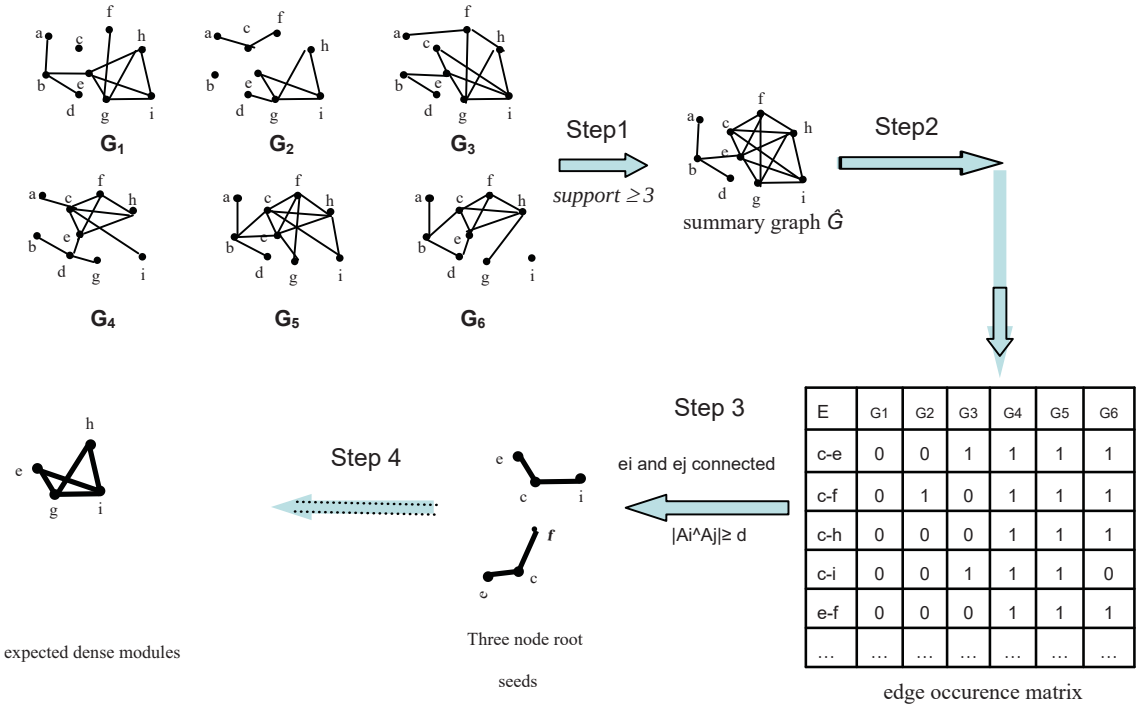
(V, E_i), a subgraph $sub(\hat{G})$ is coherent if all the edges of $sub(\hat{G})$ have support higher than k.

The problem of mining coherent dense subgraphs is formulated as follows: given a relation graph dataset,$D = \{G_1, G_2, \ldots, G_n\}$, discover subgraphs or modules g that satisfy the following two criteria simultaneously: (1) g is a densely connected subgraph of the summary graph, where density threshold is defined by γ ; and (2) g is a coherent graph, all the edges of that module satisfy the frequency threshold of occurrence in that graph network.

For the purposes of this paper, I will only be looking at the modules with γ of 0.6 or higher. The nodes with γ of 0.6 or higher has edges to at least half of the profile Attribute value other in the module, and can be deemed significant.

3.1. Algorithm

To address the above problem formulation we worked on finding smallest unit of a dense subgraph that satisfy all the two criteria simultaneously and can be solved in a much more efficient way. In this way, we got the idea of developing an algorithm for discovering dense modules using greedy approach starting from 3 node "Root". These three node "Roots"(shown in Figure 3.4) were obtained using simple quadratic Algorithm as shown in Algorithm 1. The process of creating a summary graph is shown in Figure 3.4. Pruning method was employed to extract dense summary graph from that summary graph as we can see in Figure 3.4, we maintained an edge occurence matrix which represent the binary vector for an edge across the graph network shown in Figure 3.4, we extracted the three root node and during the mining process we also used that matrix to calculate the current density of overall module shown in Figure 3.4. Like in frequent itemset mining, we created occurrence matrix, where each row contains binary representation of a certain edge in all of these graph and each column represent occurrence of all edges in the the graph network in Figure 3.4. For the first level, for every edge e_i in $\hat{E}$ in all of those dense sub graph $\hat{G}$ for various frquency threshold d, three node roots $\mathcal{R}$ were extracted using a quadratic algorithm, where each root have its own edgelist L and genelist Gen and attribute profile value A. Then using these roots we extracted dense modules using Algorithm 2 section shown in Figure 1 of "MineFrequent" Algorithm. These modules also contain a vertex set(Genes) Gen comprised of v_i,Edge list L and its current attribute profile value A and candidate Edge set $E_i.cand$ comprised of all the neighboring edges of current modules $\{E_i, E_j, E_{j+1}, \ldots, \}$ in $\hat{E}$, where j > i. As has been discussed in frequent itemset mining, this creates an enormous search space. So, we employed pruning methods and greedy approach to



E	G1	G2	G3	G4	G5	G6
c-e	0	0	1	1	1	1
c-f	0	1	0	1	1	1
c-h	0	0	0	1	1	1
c-i	0	0	1	1	1	0
e-f	0	0	0	1	1	1
…	…	…	…	…	…	…

Figure 3.4. Summary graph and dense summary graph considering minimum occurrence of each edge in a the graph network

limit our search space .

Modules have anti-monotone properties. Thus, all subsets of a module are densely connected under that density threshold and any super set of an not dense module is also non dense. We expanded module from the roots using a density calculation method based on relative number of frequency of the current module and a new edge added everytime to that module and above process has been repeated recursively.

Pruning Strategy I: As the module density γ and minimum number of occurrence,d: frequency threshold for the graph in the network $\mathcal{S}_{\mathcal{G}}$ given , the minimum occurrence for each edge in a module is the d. By eliminating edges which have occurrence less than this minimum , the algorithm significantly reduces the number of potential edges to consider in the first level. In this way, we got the dense summary graph.

Greedy Approach: we chose to follow greedy approach in every pass of a module building process In Figure 3.6, Pruning Strategy I and greedy approach for selecting module expansion branch work together to prune branches and choose Edge $\hat{E}$ from the dense summary graph $\hat{G}$.

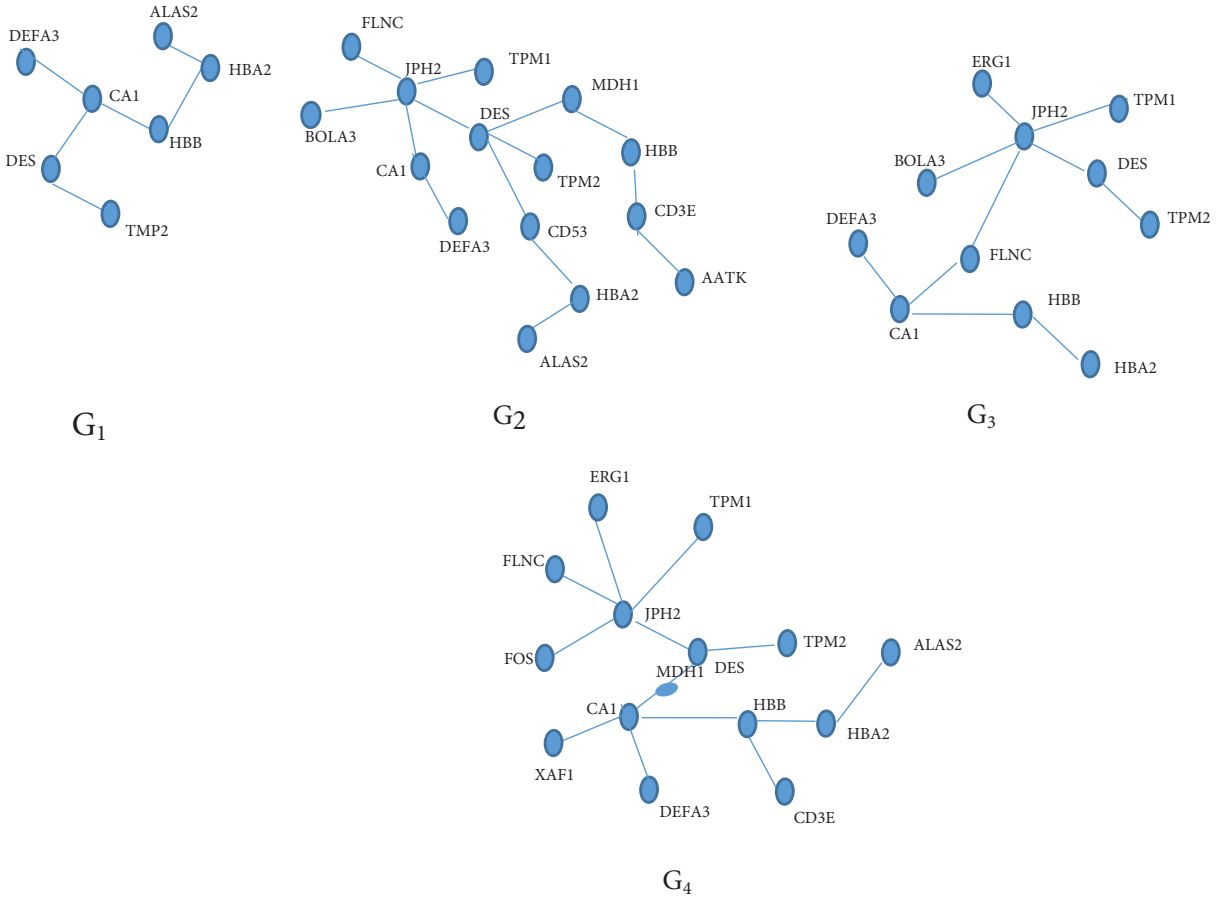


Figure 3.5. Sample co-expression graph network (partial network from four real co-expression graph)

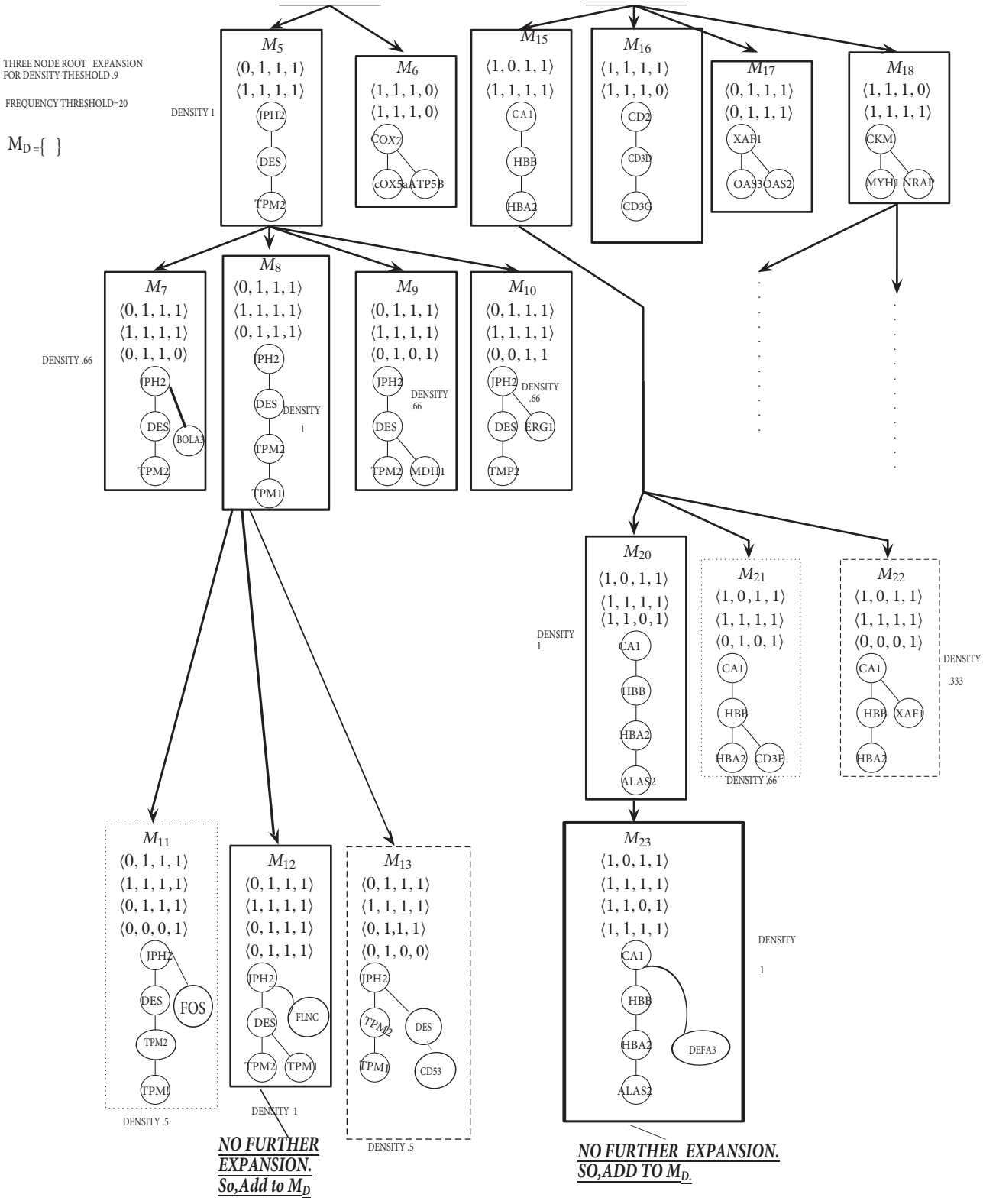


Figure 3.6. Creating an module expansion tree for a toy sample co-expression graph network using reduced Gene co-expression network shown in 3.5

Algorithm 1 Mining Three Node Root from Summary Graph

Input:

$\mathcal{S}_{\mathcal{G}}$:Co-expression Network
γ:Gamma Value of a Module
d:frequency threshold for the graph

Output:

$\mathcal{R}$: Set of three node root

1: **procedure** FINDROOT($\mathcal{S}_{\mathcal{G}}$,$\gamma$,$d$)
2: $\mathcal{R} = \emptyset$ ▷Set of three node root
3: $\mathcal{M}_{\mathcal{X}} = \emptyset$ ▷Empty Matrix for edge count
4: $\mathcal{M}_{\mathcal{F}} = \emptyset$ ▷Empty Matrix for frequent edge with count
5: V=$getVertices(\mathcal{G})$ ▷Set of Vertices in $\mathcal{G}$
6: **for each** $G_i \in \mathcal{S}_{\mathcal{G}}$:
7: **for each** $e_k \in G_i$:
8: **if** $e_k \notin \mathcal{M}_{\mathcal{X}}$:
9: **Add** e_k *in* $\mathcal{M}_{\mathcal{X}}$
10: **Set** C_k to 1
11: **elseif**
12: **Increment** C_i
13: **end elseif**
14: **end for**
15: **end for**
16: **for each** $e_i \in \mathcal{M}_{\mathcal{X}}$:
17: **if** $C_i > d$:
18: **Add** in $\mathcal{M}_{\mathcal{X}}$
19: **end if**
20: **end for**
21: **for each** $e_i \in \mathcal{M}_{\mathcal{F}}$:
22: **for each** $G_i \in \mathcal{S}_{\mathcal{G}}$:
23: **Fill** A_i of $\mathcal{M}_{\mathcal{F}}$: ▷ Attribute for every edge e_i
24: **end for**
25: **end for**
26: **for each** $e_i \in \mathcal{M}_{\mathcal{F}}$:
27: **for each** $e_j \in \mathcal{M}_{\mathcal{F}}$:
28: **if** e_i and e_j connected and $|A_i \cap A_j| > d$
29: **Add** $e_i \cap e_j$ to R
30: **end if**
31: **end for**
32: **end for**
33: **end procedure**

Algorithm 2 Dense Module Mining

$\mathcal{M}$=$getModules(\mathcal{R})$ ▷get a single three node root as Module from $\mathcal{R}$
Remove $\mathcal{M}$ from $\mathcal{R}$

procedure MINEFREQUENT($\mathcal{M}$,γ,$\mathcal{M}_\mathcal{D}$)
  ▷level 1 roots working as a seed for mining dense module
  ▷All seeds have their own edgelist L and genelist Gen and attribute profile value A
  for each $e_i \in E$:
    γ_{cur_i}=1
    if e_i not in $\mathcal{M}$'s L and e_i connected with module $\mathcal{M}$
      A_{new} =$A_{module} \cap A_i$
      calculate γ_{new}=$A_{new} \div A_{module}$
      if $\gamma_{new} > \gamma_{cur_i}$ and γ
        update γ_{cur_i} to γ_{new}
        keep update of the current edge e_i with γ_{cur_i}
      end if
    end if
  end for
  update Edgelist L and Genelist Gen of module $\mathcal{M}$ by adding edge e_i
  if update not possible and module $\mathcal{M} \notin \mathcal{M}_\mathcal{D}$:
    Add $\mathcal{M}$ to $\mathcal{M}_\mathcal{D}$
    return
  Minefrequent($\mathcal{M}$,γ,$\mathcal{M}_\mathcal{D}$)
end procedure

3.2. Pseudo-code

Algorithm 1 shows the main pseudo-code of the Mining maximally frequent dense module (MineFrequent) approach which first find frequent three node roots from edge occurrence matrix.The algorithm takes as input a set of co-expression Networks $\mathcal{S}_\mathcal{G}$, the module density γ, summary graph $\hat{\mathcal{S}}_\mathcal{G}$ build from all of these networks, using pruning strategy I from this summary graph a dense summary graph $\hat{G}$ was created for every frequency threshold d in the graph network. Edge Occurrence Matrix $\mathcal{M}_\mathcal{X}$ was build from that dense summary graph. The algorithm outputs $\mathcal{R}$,set of all three Node seeds. First, a summary graph was created from set of co -expression network $\mathcal{S}_\mathcal{G}$, using that summary graph we created edge occurrence matrix $\mathcal{M}_\mathcal{X}$ or attribute profile for every edge e_i. From the Edge Occurrence Matrix $\mathcal{M}_\mathcal{X}$, We extracted all the three node root, just by checking whether these two edges are connected or not and they appeared in at least d or more graph by doing logical **And** for their corresponding Attribute row A_i.

Algorithm 2 shows the maximal dense module extraction from each seed of $\mathcal{R}$. Like travers-

ing the frequent itemset set enumeration tree, each three node root's candidate set is combined with its edgeset and genelist to expand that seed or module for further expansion. After getting the data structure ready , for each γ value varying from .6 to .9 we extracted every seed $\mathcal{M}$ from set of seed R with it genelist and edgelist (lines 2-4).After that, we considered all its neighboring edge set $E_i.cand$ to be considered within that density threshold and update that module. For each seed, we get only one maximally dense module. If we already have that module generated from other seed we consider only one (lines 15-18). If the module is still updating, we then considered it for further expansion(line 19).

4. EXPERIMENTS

4.1. Description of Data-set

To determine if the algorithm is efficient in finding maximal dense module, its efficiency and effectiveness are tested on a real-world dataset. A dataset is constructed from from GTEx data, created a platform called Genetic Network Analysis Tool lets users instantly search and visualize human genetic networks dataset [14] . Tissue-specific mechanisms of control may be captured by co-expression networks, in which two genes are connected if their expression levels are correlated across a set of individuals. In such a setting, genetic or environmental differences across individuals serve as small perturbations to the underlying regulatory network, resulting in correlation between genes' expression levels that are consistent with regulatory relationships. Co-expression networks provide better insight into cellular activity as genes that are co-expressed often share common functions by reducing noise and removing spurious links [16].

The Genotype-Tissue Expression (GTEx) consortium dataset [8] provides an opportunity to study such co-expression networks for an unprecedented number of human tissues simultaneously. However, many of the profiled tissues have fewer than a dozen samples, too few to accurately infer the tens of millions of parameters that would define a co-expression or regulatory network. One solution would be to combine all available samples and learn a single consensus network for all tissues, but this would offer no insight into tissue-specificity. On the other hand, inferring each network independently ignores tissue commonalities: tissue networks share far more links than would be expected by chance, and learning links across multiple tissues is less noisy than learning links using a single tissue, even when using the same number of total samples [16].

Figure 4.1 represents the summary graph of the GTEx network. It contains 35 distinct human tissues, 9998 genes(nodes) , 5 million links.

4.2. Design of Experiment and Results

The algorithm for detecting maximal dense module had been run with varying frequency threshold d to create dense subgraph and then three node roots were extracted from these dense subgraphs.We got different numbers of root for different frequency threshold , d.Then, using all these roots for specific d, we chose varying density threshold γ or module density to be mined from

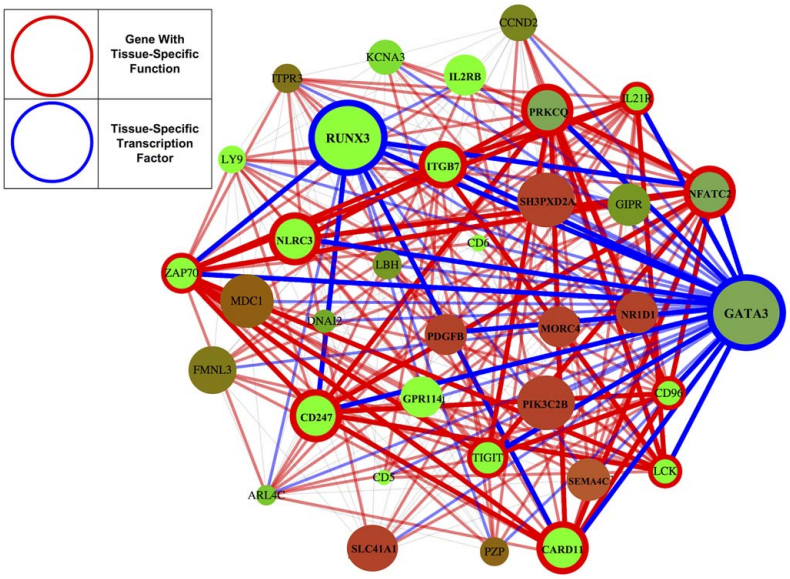


Figure 4.1. The GTEx dataset.

the graph network. Module density γ had been varied from 0.6 to 0.9 in .1 increments based on the running time for specific d ; The minimum size of the maximal module or average number of gene in each module varied from five to eleven.The minimum number of links in all of these modules varied from four to eleven in varying density increments. In Table 4.1 , we can see data found as output of Algorithm 1 for various frequency threshold, d , different dense subgraph considering different threshold. This table (table 4.1 gives us valuable information about the running time of first portion of MineFrequent approach or Algorithm 1. We also get valuable insight about the extracted three node roots and our new approach.

Table 4.1. Seed extraction From Dense Sub graph

Freq Thres	Total Seed (Algorithm 1)	Time(s)	Dense Module(Algorithm 2)
18	3678	23980	2268
20	2097	10890	1402
22	1224	7980	833
24	760	5103	524

For Example, for frequency threshold 18, we got 3678 three node roots running Algorithm 1 of MineFrequent approach while it took 23980 second to generate such output and using Algorithm 2 we found 2268 dense modules out of these 3678 starter three node roots. In Figure 4.2, we can see

the performance of Algorithm 1 of MineFrequent approach, which is non-exponential as we wanted in the beginning of our paper and it looks like almost quadratic. In Figure 4.3, we can also see the number of final modules extracted as output of Algorithm 2 of this approach reduced exponentially with the increase of frequency threshold, d. These final modules were expanded starting from seeds as we got them as the output of Algorithm 1.

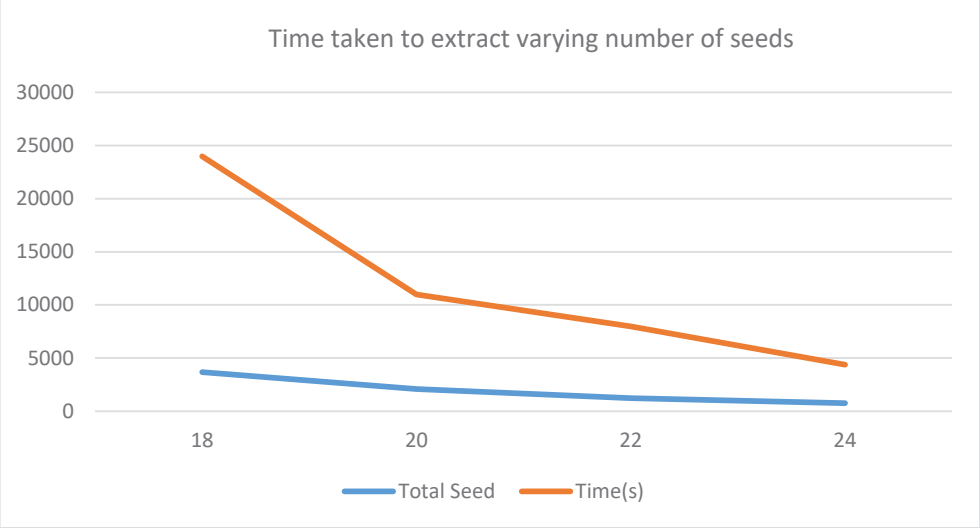


Figure 4.2. Improved non-exponential almost quadratic runtime of Algorithm 1 for extracting roots

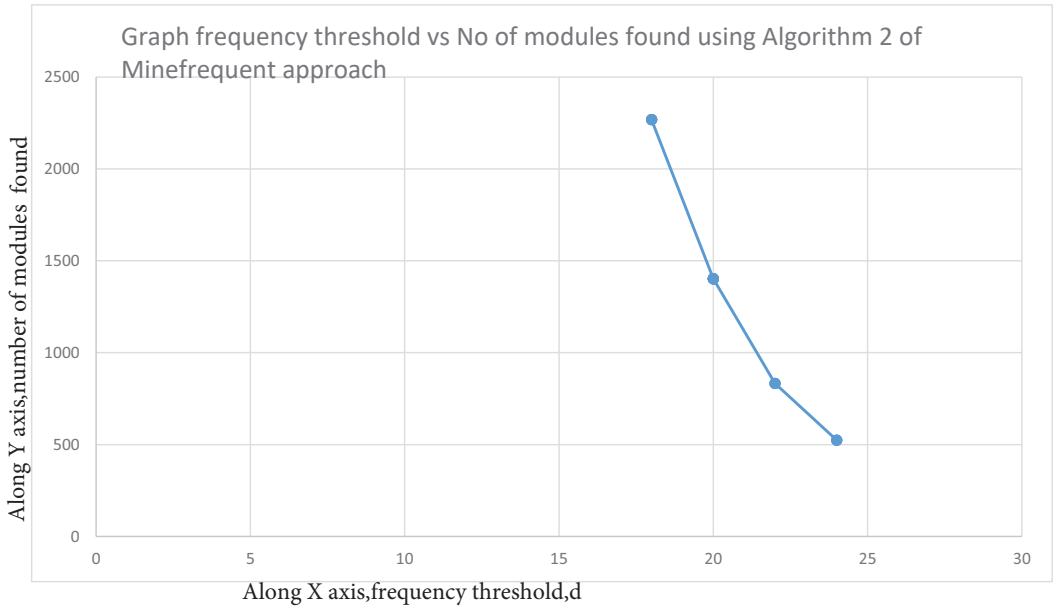
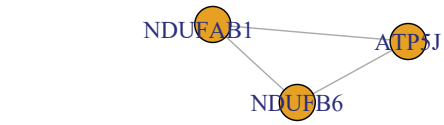


Figure 4.3. Number of Module extracted from graph network vs frequency threshold of dense sub graph using Algorithm 2 of Minefrequent approach



Three node seed extracted using frequency threshold,d =20 from 35 co-expression network

Figure 4.4. Example of a three node module extracted using Algorithm 1

Table 4.2 shows the results of running time of the Algorithm 2 of MineFrequent approach for the roots generated from Algorithm 1 under different d using specific density threshold, γ. In the first row of table 4.2, we can see the running time of Algorithm 2 for d= 18 and $\gamma = 0.6$. The longest running time that the algorithm took to complete, when the setting of $\gamma = 0.6$ and d= 18. This took almost 32600 seconds. However, if either γ was increased, the running time was dramatically reduced, module **AvgGeneNum** also reduced. The next longest running time was obtained under the settings of $\gamma = 0.7$ and d= 18 . This took around 24600 seconds.

Another thing to note here is that the dense modules found were mostly highly connected with each member of the modules and found in at least 20 graphs among 35 graphs. That is to say, their average density were varying from .849 to .956 . The density of a graph is $|E|/(|V|\times|V|-1)/2$,

G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}	...	G_{33}	G_{34}	G_{35}	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	0	1	1	1	1	1	1	1	1	...	1	1	1	15
1	1	1	0	1	0	1	1	1	1	...	1	1	1	144
1	1	1	1	1	1	1	1	1	0	...	0	1	1	1192
1	1	1	1	1	1	1	0	1	1	...	0	0	0	1193
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	1	1	1	0	1	1	...	1	1	1	1209
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	0	1	0	1	1	1	0	1	1	...	1	1	1	1267
1	1	1	1	1	0	1	1	1	0	...	1	1	1	1269
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	0	1	0	1	0	1	0	1	0	...	0	0	0	$\mathbf{A_{Mfinal}}$

Figure 4.5. Module density calculation starting from a seed shown in figure 4.4 having Genes NDUFAB1,NDUFA4,NDUFA9,ATP5J,ATP5B,COX7B,ATP5G3 from ATTRIBUTE profile value A_i and A_M Edge Occurrence Matrix $\mathcal{M}_{\mathcal{X}}$ has been shown in this figure for all edges in the module.module density is .909090.

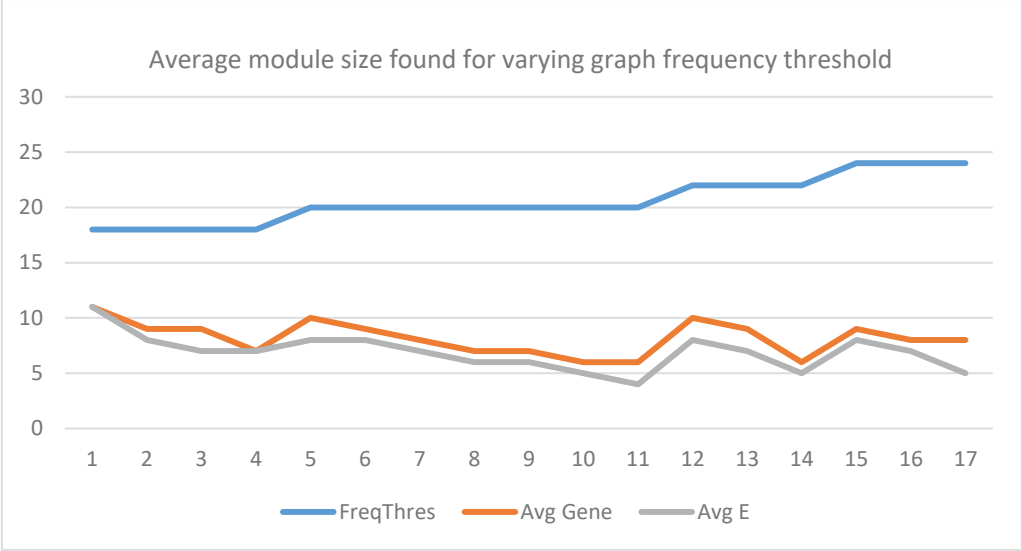


Figure 4.6. Average size of Modules extracted from graph network in terms of nodes(genes) and edges vs frequency threshold of dense sub graph using Algorithm 2 of Minefrequent approach

or the number of edges in a graph over the maximum possible number of edges within the graph. For most of the settings, the average density of the module found all have much higher density than the density threshold. Another thing to note is that when we increased the density threshold of module, A_{module} also increased . That means, when we increase the density threshold ,we found much more dense and frequent module.

In Figure 4.6, we can see, average size of the modules extracted using Algorithm 2 of this

Table 4.2. Analysis of Mining Maximal Dense Module Found with GTEx Data

FreqThres	*Density*	**Time(s)**	**Avg Gene**	**Avg E**	**Module**	$\overline{Density}$
18	.60	32600	11	11	2268	0.85912
18	.70	24600	9	8	2268	0.88005
18	.80	18786	9	7	2268	0.90123
18	.90	12890	7	7	2268	0.9492
20	.60	8640	10	8	1402	0.849835
20	.65	6664	9	8	1402	0.8590
20	.70	5956	8	7	1402	0.8690
20	.75	5680	7	6	1402	0.879177
20	.80	5280	7	6	1402	0.8930
20	.85	4945	6	5	1402	0.91115
20	.90	4790	6	4	1402	0.93356
22	.60	11760	10	8	833	0.8531
22	.70	9680	9	7	833	0.8821
22	.90	4380	6	5	833	0.9400
24	.60	8970	9	8	524	0.8642
24	.70	6790	8	7	524	0.8997
24	.90	5890	6	5	524	0.9564

approach in term of nodes(genes) number and edges or links number with respect to frequency threshold, d. we can hardly see any regular pattern like number of modules or size of module vs d or running time ,t.

In Figure 4.9, we can see, the running time of Algorithm 2 of MineFrequent approach using almost 17 data point found for different d and varying module density threshold γ. From the graph, we can conclude that , running time of Algorithm 2 of MineFrequent approach clearly non-exponential and it looks like almost quadratic.

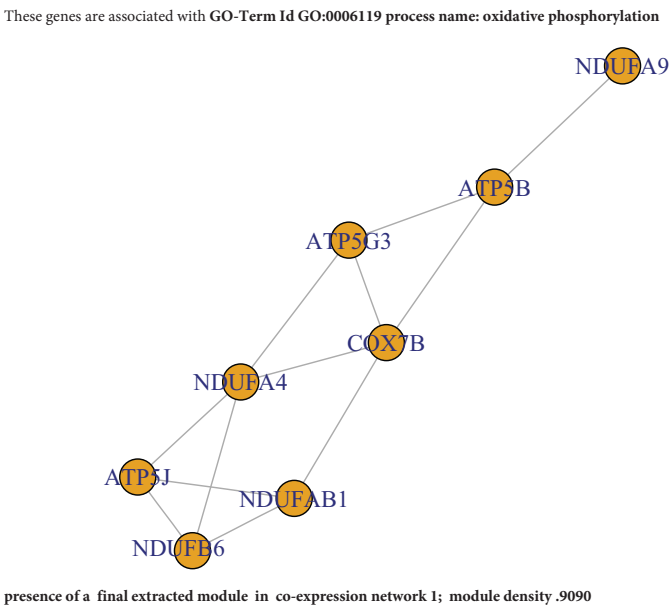


Figure 4.7. Example of a real module extracted using MineFrequent approach shown in Network 1 by expanding root shown in Figure 4.4

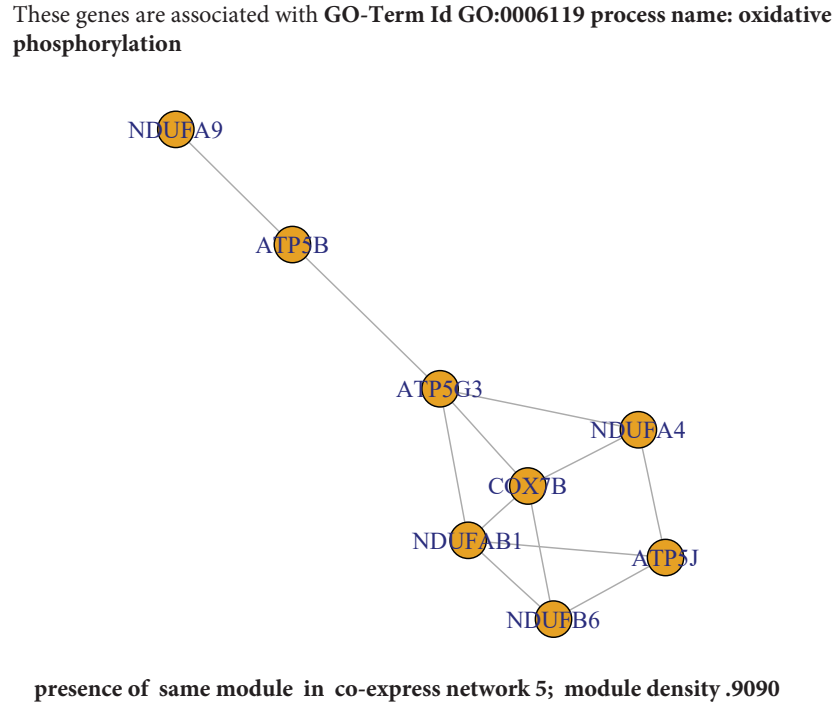


Figure 4.8. Same module appeared in Network 5

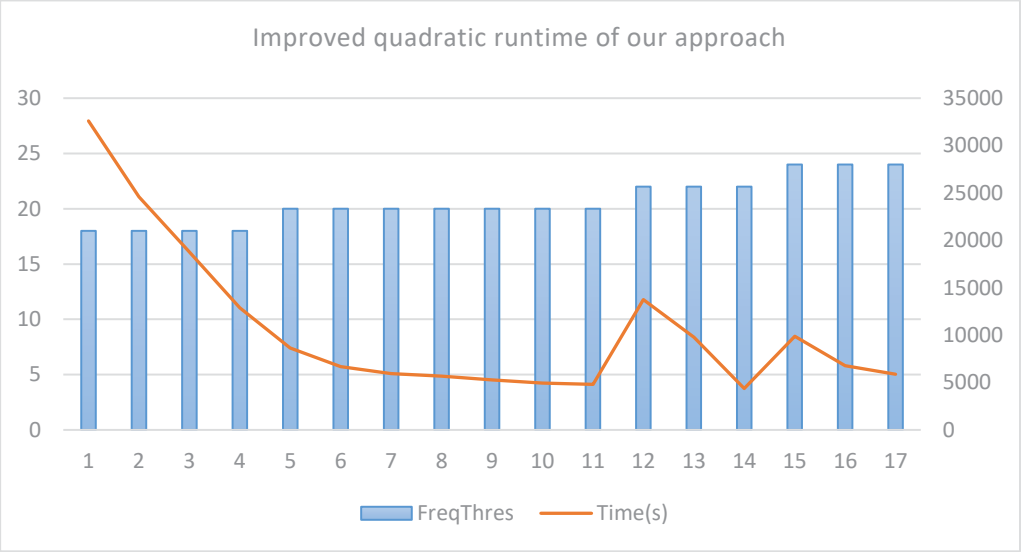


Figure 4.9. Running time of algorithm 2 of MineFrequent approach vs frequency threshold

4.3. Enrichment Analysis for Dense Module

Gene Set Enrichment Analysis (GSEA) is a computational method that determines whether an a priori defined set of genes shows statistically significant, concordant differences between two biological states. It is also known as functional enrichment analysis. This term can also be defined as a method to identify classes of genes or proteins that are over-represented in a large set of genes or proteins, and may have an association with specific phenotypic state (i.e disease,cancer). The method uses statistical approaches to identify significantly enriched or depleted groups of genes. Microarray and proteomics results often identify thousands of genes which are used for the analysis Ranking functional categories based on co-occurrence with sets of genes in a gene list can rapidly aid in unraveling new biological processes associated with cellular functions and pathways. Researchers performing high-throughput experiments that yield sets of genes (for example, genes that are differentially expressed under different conditions) often want to retrieve a functional profile of that gene set, in order to better understand the underlying biological processes. This can be done by comparing the input gene set to each of the bins (terms) in the gene ontology – a statistical test can be performed for each bin to see if it is enriched for the input genes.

As in the begining of our paper, we said genes expressed in multiple co-expression network with high d were extracted as dense module as output of our MineFrequent process and these

same set of genes appeared in multiple co-expression network build within same specific context or microarray data set and all these genes appeared as a set representing the context or environment of expressing genes, these modules are highly likely to be enriched. While the completion of the Human Genome Project gifted researchers with an enormous amount of new information, it also left them with the problem of how to interpret and analyze the incredible amount of data. In order to seek out genes associated with diseases, researchers utilized DNA micro arrays, which measure the amount of gene expression in different cell, GSEA method focuses on the changes of expression in groups of genes, and by doing so, this method resolves the problem of the undetectable, small changes in the expression of single genes [19].

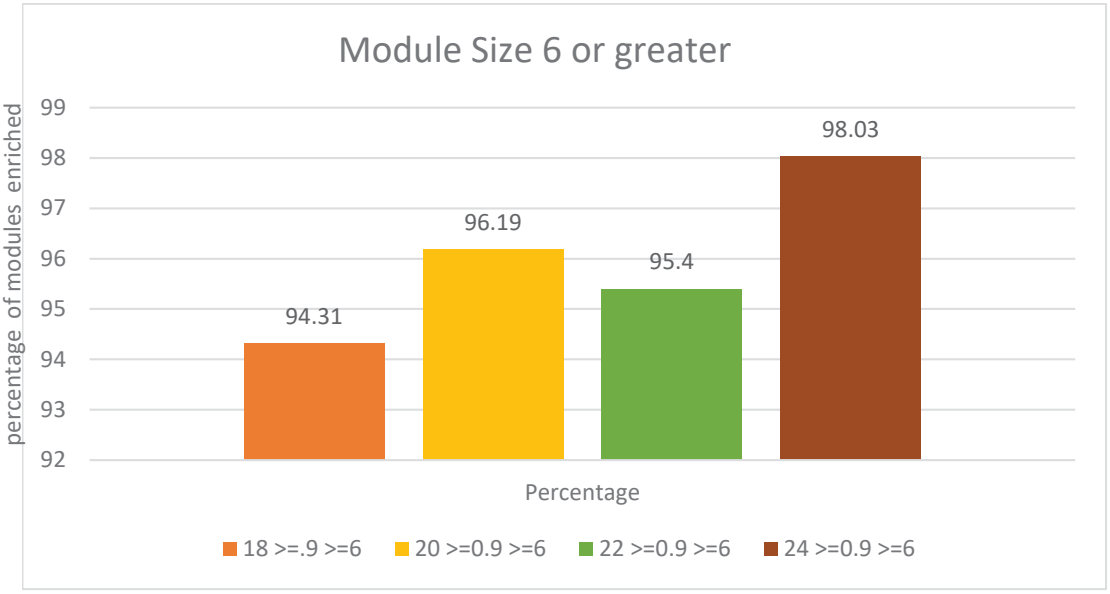


Figure 4.10. Percentage of module enriched while considering module size 6 or greater

In the method that is typically referred to as standard GSEA, there are three steps involved in the analytical process [2].These steps are summarized below:

1. Calculate the enrichment score that represents the amount to which the genes in the set are over-represented at either the top or bottom of the list. This score is a Kolmogorov–Smirnov-like statistic [2]

2. Estimate the statistical significance of the Enrichment Score (ES). This calculation is done by a

phenotypic-based permutation test in order to produce a null distribution for the ES [2].

3. Adjust for multiple hypothesis testing for when a large number of gene sets are being analyzed at one time. The enrichment scores for each set are normalized and a false discovery rate is calculated [2].

ClusterProfiler [24] allows investigators to sort gene categories from dozens of annotation systems, this is an R package for comparing biological themes among gene clusters. Sorting can be based on the number of genes within each category. Any given gene is associating with a set of annotation terms. If genes share similar set of those terms, they are most likely involved in similar biological mechanisms. The algorithm tries to group those related genes based on the agreement of sharing similar annotation terms by hypergeometric ***P-value*** < 0.01. Using clusterprofiler, we tried to annotate all the dense modules found from our "MineFrequent" approach. In Table 4.3, we can see the percentage of final modules were enriched using clusterprofiler while considering modules size six or grater.In Table 4.4 enrichment results are shown for all the modules extracted using Algorithm 2 or modules have size three or greater than that.We can say by comparing these tables that modules with high density and greater size are highly likely to be enriched. In table 4.5, top four GO(Gene Ontology) process with GO Id has been shown while performing enrichment of modules extracted using our MineFrequent method.

Table 4.3. Percentage of Module Enriched when considering module density more than .9 and module size more than 6 for various d= 18,20,22,24

Freq Thres	M density $\geq$	M Size $\geq$	Percentage
18	.9	6	94.31
20	.9	6	96.19
22	.9	6	95.4
24	.9	6	98.03

Table 4.4. Percentage of Module Enriched when considering module density more than .9 and module size more than 3 for various d= 18,20,22,24

Freq Thres	M density $\geq$	M Size $\geq$	Percentage
18	.9	3	82.58
20	.9	3	84.30
22	.9	3	83.19
24	.9	3	86.06

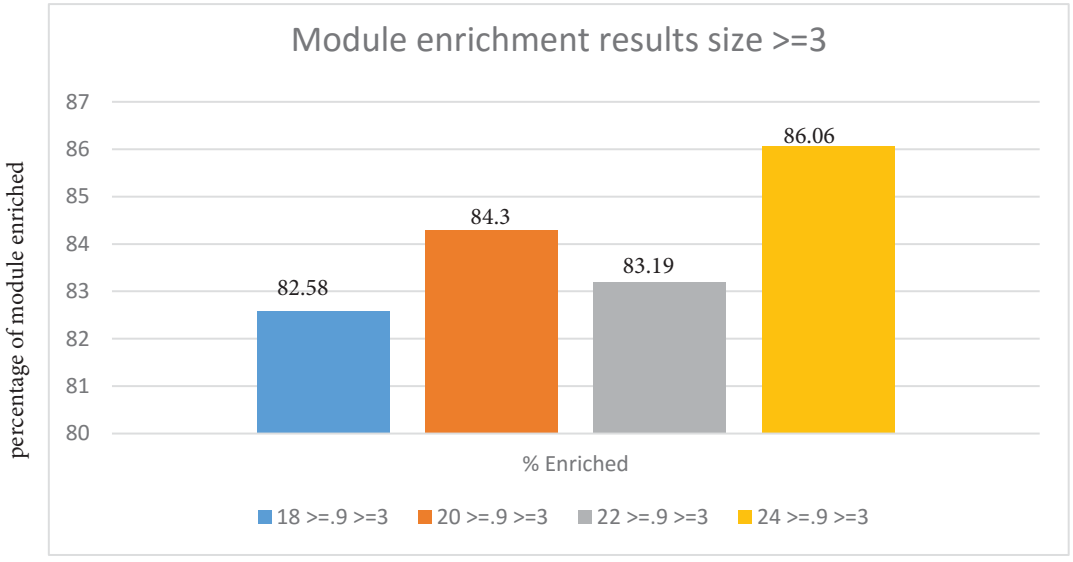


Figure 4.11. Percentage of module enriched while considering module size 3 or greater

Table 4.5. Top 4 GO(Gene Ontology) processes with their GO:ID module density $\geq .9$ and module size ≥ 6 for various d= 18,20,22,24

Freq Thres	Go Id	Go Process
18	GO:0006613	cotranslational protein targeting to membrane
	GO:0006614	SRP-dependent cotranslational protein targeting to membrane
	GO:0019058	viral life cycle
	GO:0045047	protein localization to endoplasmic reticulum
20	GO:0019058	viral life cycle
	GO:0006612	protein targeting to membrane
	GO:0006613	cotranslational protein targeting to membrane
	GO:0019083	viral transcription
22	GO:0000184	cellular nitrogen compound catabolic process
	GO:0006612	protein targeting to membrane
	GO:0006613	cotranslational protein targeting to membrane
	GO:0000956	nuclear-transcribed mRNA catabolic process
24	GO:0006613	cotranslational protein targeting to membrane
	GO:0000956	nuclear-transcribed mRNA catabolic process
	GO:1901605	alpha-amino acid metabolic process
	GO:0019439	aromatic compound catabolic process

5. CONCLUSION AND FUTURE WORK

In conclusion, using a summary graph and building a dense summary graph based on different density threshold to find meaningful module in networks helped find interesting interactions among genes. Our developed algorithm, MineFrequent, to efficiently mine coherent dense subgraphs across massive biological networks; In comparison with previous approaches, is scalable in the number and the size of the networks to mine, adjustable in terms of exact or approximate coherent pattern mining, and extendable to weighted and directed networks. However, there are a few weaknesses that should be addressed in future work. One of them is that as modules do not have an anti-monotone property, the algorithm must traverse large areas of the set enumeration tree to find modules and another is genes among various modules intersects. Although we have a few pruning strategies, they do not seem to be enough. This can lead to very long running times.

In future,we are planning to integrate simulated annealing approach with anti-monotone property so that no overlapping module produced and maintain anti-monotone property that can prevent generating further branch that could lead to the existing module. For producing dense subgraph we can employ better strategy to get more precise edgeoccurence matrix ,so that, the overall process get more efficient result.

The discovered network modules can be used in a variety of biological applications, e.g., predict the functions of unknown genes, construct the transcription modules, and infer the potential protein assembly mechanisms.

Despite these limitations, the modules extracted by our approach can aid biologists in making sense of the large amounts of information often produced by highthroughput experiments. Mapping this information onto functional modules rather than large pathways makes highthroughput experiments easier to understand. Furthermore, our representation of the modules in a functional network can help biologists trace the transfer and integration of information and interaction between and among modules, and can lead to experimentally verifiable hypotheses.

REFERENCES

[1] Rakesh A and Ramakrishnan S. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, 1994.

[2] Subramanian A, Tamayo P, and Moothka V. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 2005.

[3] Barabasi AL and Oltvai ZN. Network biology: understanding the cell's functional organization. *Nat Rev Genet*, 5:101–13, 2003.

[4] Gasch AP and Eisen MB. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol*, 3:RESEARCH0059, 2002.

[5] Victor E, Ning R, Ruoming J, and Charu A. A survey of algorithms for dense subgraph discovery. In Aggarwal CC and Haixun W, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 303–336. Springer US, 2010.

[6] Haiyan H and Zhou XJ. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 1:2–3, 2005.

[7] Haiyan H and Zhou XJ. Systematic discovery of functional modules and context-specific functional annotation of human genome. *Bioinformatics*, 23:i223–i224, 2007.

[8] Lonsdale J and Thomas J. The genotype-tissue expression (gtex) project. *Nature Genetics*, 45(6):580–585, 2013.

[9] Mehmet K, Yohan K, Shankar S, Wojciech S, and Ananth G. Detecting conserved interaction patterns in biological networks. *Journal of Computational Biology*, 13(7):1299–1322, 2006.

[10] Xiaoli L, Min W, Chee KK, and See KN. Computational approaches for detecting protein complexes from protein interaction networks: a survey. *BMC Genomics*, 11(Suppl 1):S3, 2010.

[11] Peri LJ and Navarro MK. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Res*, 13:2363–2903, 2003.

[12] Peri LJ and Navarro MK. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Res*, 13:2363–2903, 2003.

[13] Kuramochi M and Karypis G. Finding frequent patterns in a large sparse graph. *2004 SIAM Data Mining Conference*, 2004.

[14] Emma P, Alexis B, and Sara M. Sharing and specificity of co-expression networks across 35 human tissues. *PLoS Comput Biol*, 11(5):e1004220, 2015.

[15] Sharan R, Ideker T, Kelley B, Shamir R, and Karp RM. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, RECOMB '04, pages 282–289, 2004.

[16] Piro RM and Ala U. An atlas of tissue-specific conserved coexpression for functional annotation and disease gene prediction. *European Journal of Human Genetics*, 19(11):1173–1180, 2011.

[17] Calvano SE and Xiao W. A network-based analysis of systemic inflammation in humans. *Nature*, 437:1032–1037, 2005.

[18] Andrzej T. Enumerated sets. *Journal of Formalized Mathematics*, 1, 1989.

[19] Moothka V. Pgc-1a-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetics. *Nature Genetics*, 34(3):267, 2003.

[20] Oron V, Oded M, and Tomer S. Associating genes and protein complexes with disease via network propagation. *PLoS Comput Biol*, 6(1):e1000641, 2010.

[21] Zhou XJ and Han J. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Mining Closed Relational Graphs with Connectivity Constraints. Proceedings of the International Conference on Data Engineering*, 2005.

[22] Zhou XJ and Kao MC. Transitive functional annotation by shortest-path analysis of gene expression data. *Proc. Natl Acad. Sci. USA*, 99:12783–12788, 2003.

[23] Zhou XJ and Michael SW. Systematic discovery of functional modules and context-specific functional annotation of human genome. *Bioinformatics*, 23:i222–i229, 2007.

[24] Guangchuang Y, Yanyan H, and Qing-YH. Clusterprofiler: an r package for comparing biological themes among gene clusters. *OMICS: A Journal of Integrative Biology*, 16(5):284–287, 2012.