CLASSIFYING GENE COEXPRESSION NETWORKS USING DISCRIMINATIVE PATTERN MINING

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

By

Bassam M M Qormosh

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

Major Department:
Computer Science

September 2016

Fargo, North Dakota

NORTH DAKOTA STATE UNIVERSITY

Graduate School

Title

CLASSIFYING GENE COEXPRESSION NETWORKS USING DISCRIMINATIVE PATTERN MINING

By

Bassam M M Qormosh

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:

Saeed Salem
Chair

Simone Ludwig

Mukhlesur Rahman

Approved:

9-28-2016
Date

Brian M. Slator
Department Chair

ABSTRACT

Several algorithms for graph classification have been proposed. Algorithms that map graphs into feature vectors encoding the presence/absence of specific subgraphs, have shown excellent performance. Most of the existing algorithms mine for subgraphs that appear frequently in graphs belonging to one class label and not so frequently in the other graphs. Gene coexpression networks classification attracted a lot of attention in the recent years from researchers in both biology and data mining because of its numerous useful applications. The advances in high-throughput technologies that provide an easy access to large microarray datasets necessitated the development of new techniques that can scale well with large datasets and produce a very accurate results. In this thesis, we propose a novel approach for mining discriminative patterns. We propose two algorithms for mining discriminative patterns and then we use these patterns for graph classification. Experiments on large coexpression graphs show that the proposed approach has excellent performance and scales to graphs with millions of edges. We compare our proposed algorithm to two baseline algorithms and we show that our algorithm outperforms the baseline techniques with a very high accurate graph classification. Moreover, we perform topological and biological enrichment analysis on the discriminative patterns reported by our mining algorithm and we show that the reported patterns are significantly enriched.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor, Dr. Saeed Salem. His patience, motivation, and guidance helped and motivated me in all the time of research and writing of this thesis. I would also like to thank my committee members, Dr. Simone Ludwig and Dr. Mukhlesur Rahman, for taking the time to evaluate my work and helping me to graduate from the University. I also would like to thank North Dakota State University and the Department of Computer Science for offering me the opportunity to study at such a great school. Last but not the least, I would like to thank my family for supporting me throughout writing this thesis and my life in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Graphs have emerged as natural data structures to model complex chemical and biological systems, e.g., chemical compounds, protein 3D structure, biological pathways, coexpression networks, and protein-protein interaction networks. Researchers in graph learning and mining have proposed algorithms that have been applied in various applications, including chemical compound classification, identifying drug targets, gene function prediction, subnetwork biomarkers [14].

The huge advances in genome sequencing and high-throughput technologies that provided easy access to large micro-array datasets led to the popularity of coexpression networks in biology. Coexpression networks analysis provides an easy and reliable way to discover new genes functions and understand complex biological processes. The construction of coexpression networks is quite simple where genes expression profiles across tissues samples are used to create a network of genes where two genes are connected if they have similar expression profiles [20].

Research in coexpression networks analysis showed that genes that are connected in the network usually indicate that these genes are functionally related. Stuart et al. [20] used microarray datasets to create coexpression network where they applied k-means clustering to discover large components with highly-interconnected meta-genes. They showed that many of the components were enriched for meta-genes involved in similar biological processes.

Coexpression networks analysis usually involves very large datasets which requires efficient techniques to process and discover useful information with a reasonable amount of time. This problem attracted data mining researchers to utilize and propose data mining techniques for genes coexpression networks analysis. One of the most popular techniques that have been utilized in coexpression networks analysis is coexpression graph classification [18].

Graph classification attracted a lot of attention in recent years because of the wide range of applications in which it can be applied. For example, in chemical Informatics, chemical compounds are often represented as graphs; graph classification algorithms can be applied on these graphs to predict for example the toxicity of these compounds [18]. Coexpression networks classification involves a dataset of graphs labeled with a class label, for example a set of graphs annotated with a phenotype and another set of graphs that serves as a background class. The task is to classify graphs that are not labeled with the class label using information extracted from the coexpression graphs dataset. The application of this techniques opens the door to a reliable and efficient way to functional annotation of unknown genes and many other useful discoveries.

1.1. Contribution

In this thesis, we propose two algorithms for coexpression networks classification using discriminative subgraphs mining. We use microarray datasets to create the coexpression networks. Following that, we mine discriminative patterns from the coexpression networks and use these patterns as features to train a decision tree classifier. We propose two different objective functions to measure discriminative power of a subgraph then compare our results with two baseline algorithms. In the first algorithm we show that the use of proposed OR function outperform the traditional AND function that is traditionally used to calculate the discriminative power of a pattern. In the second algorithm, we utilize an antimonotone function to calculate the discriminative power of a pattern which helps prune the search and minimize the search space. Our result outperform the baseline algorithms and present very accurate classification results.

1.2. Thesis Overview

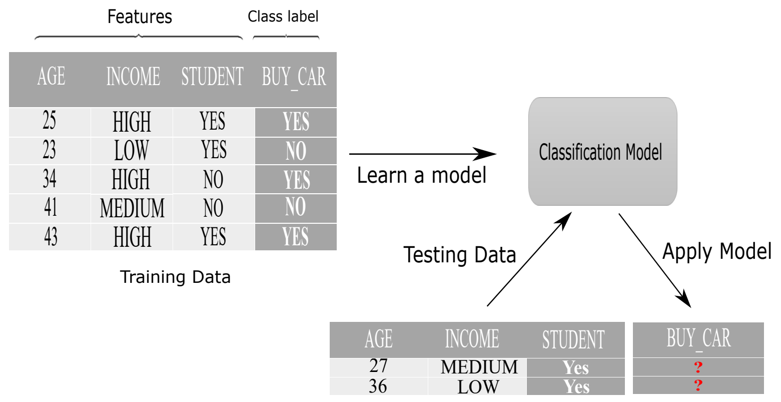
The remainder of this thesis is organized as follows, In Section 2, we present some important and related topics to our work. In Section 3, we introduce our first proposed algorithm including the problem description and the details of the proposed algorithm followed by the experiment details and results. In Section 4, we introduce our second proposed algorithm starting with important definitions and problem description along with the details of the proposed algorithm then move to the experiment details and results of the second algorithm. Finally, Section 5, outlines the conclusion and possible future work.

2. RELATED WORK

In this section, we present the related work starting with a brief background about important topics related to our work including classification, frequent pattern mining, coexpression networks construction and analysis. Following that, we present the current related research work.

2.1. Classification

Classification is a data mining technique that is used to predict the label of unlabeled data instance. Usually we have a labeled dataset and the task is to predict the label of unlabeled data instance using information from the labeled dataset. Many techniques and algorithms have been proposed to do data classification including Decision Trees, Bayesian Networks and k-Nearest Neighbor classifiers. Figure 2.1 shows how a classification model is created. The process starts by using training data with known class labels to learn a classification model using some learning algorithm. Once the model is created, we apply the model on testing data to predict the class label.



AGE	INCOME	STUDENT	BUY_CAR
25	HIGH	YES	YES
23	LOW	YES	NO
34	HIGH	NO	YES
41	MEDIUM	NO	NO
43	HIGH	YES	YES

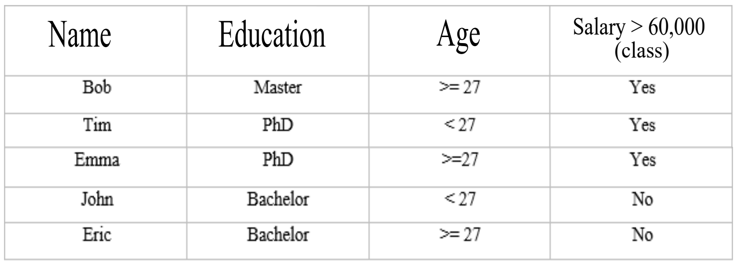
AGE	INCOME	STUDENT	BUY_CAR
27	MEDIUM	Yes	?
36	LOW	Yes	?

Figure 2.1. Classification model example.

2.1.1. Decision Trees

Decision Trees uses a tree structure to classify data instances. In a decision tree, the leaves are class labels internal nodes are features on which the instances are tested and branches are conjunctions of features that lead to the class labels [19]. Starting from the root node, the data instances are split based on the feature in the node and the process continues until it reaches the class labels at the leaves.

The feature at the root node is chosen to be the feature that best divide the data [17]. Many research works have been done to find a way to choose the feature that best divide the data instances such as information gain [10] and gini index [1]. Choosing the best feature at the root node can help the tree converges quickly to a subset of instances of the same class label.



Name	Education	Age	Salary > 60,000 (class)
Bob	Master	>= 27	Yes
Tim	PhD	< 27	Yes
Emma	PhD	>=27	Yes
John	Bachelor	< 27	No
Eric	Bachelor	>= 27	No

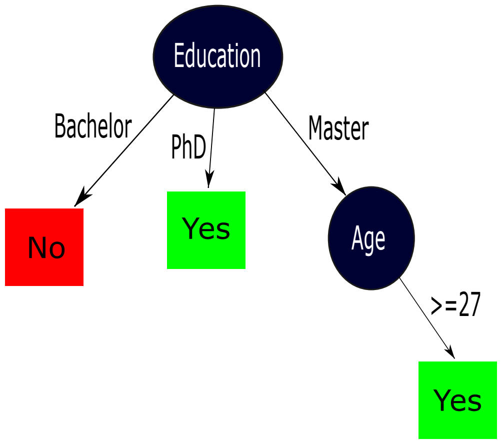


Figure 2.2. Decision tree classification example.

An example of how the decision tree work is presented in Figure 2.2 where company data about employees is used to decide how much the salary of a new employee is expected to be. The feature at the root is the Education, based on this attribute the data is divided into three sets: Bachelor, Master, or PhD. From the decision tree we can see that every employee with a bachelor degree will have a salary less than $60,000 and every employee with PhD degree will have a salary more than $60,000 regardless of the age attribute. For employees with Master degree, the age attribute is used to decide if the employee will have a salary greater than $60,000 or not. From this decision tree, the salary for any new employee can be predicted easily.

2.1.2. Bayesian Network

Bayesian Network classifier is a probabilistic model where a set of random variables and their conditional dependencies are represented as a directed acyclic graph (DAG) [19]. In the Bayesian Network, nodes represent the random variables and the edges represent the conditional dependencies between the random variables. Moreover, each node in the network is associated with a table that represent the conditional distribution of the variable given its parents [17].

To predict the class label of a data instance, the Bayesian network is used to model the features and class labels of the labeled dataset. Using the network, Bayesian network classifier will predict the class label with the highest conditional probability [5] as following :

$$P(\boldsymbol{x}, C) = P(C|Parent(C)) \prod_{i=1}^{n} P(x_i|Parent(x_i))$$

Where $\mathbf{x} = (x_1, \cdots, x_n)$ is the data instance attributes and C is the class label. The predicted class label will be the class label with the highest probability.

2.1.3. k-Nearest Neighbor

In a k-Nearest Neighbor classifier, the data instances are classified based on their nearest neighbors. An instance's class label is determined by finding the class labels of its closest k neighbors [2]. The algorithm simply find the k nearest neighbors class labels where k is the number of neighbors, then it will predict class label of the data instance same as the label of the majority of the neighbors (majority vote).

Before running the algorithm, two metrics must be determined; the distance and the value of k. There are different distance metrics can be used depending on the type of the dataset, for example: the Euclidean distance, the Manhattan distance, or any other form of the general distance measure known as Minkowski distance. The second metric that need to be determined is the value of k. A large value of k can result in wrong classification since far neighbors will be included in the classification. On the other hand, a small value can produce wrong results when there are some noisy data instances close to the data instance.

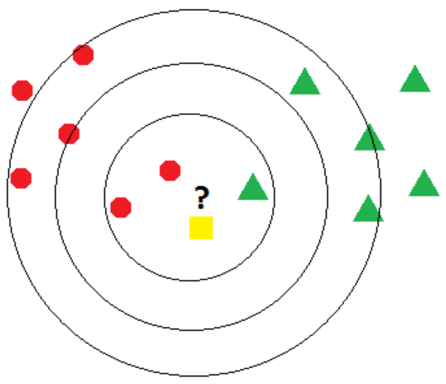


Figure 2.3. K-NN classification example with k = 3.

Figure 2.3 presents an example of K-NN classifier used to decide the class label of a shape. Using k = 3, the closest three shapes are two circles and triangle. Since the majority of the three neighbors are circles, then the shape will be labeled as a circle.

One disadvantage of the K-NN classifier is that it needs to be calculated for every different data instance. Unlike other classifiers like decision trees that builds a model one time and use that model for every data instance, the K-NN classifier does not build any model and the calculation is done for every instance separately, this why it is called *Lazy Learning* classifier.

2.2. Frequent Pattern Mining

Frequent pattern mining is one of the most important techniques in data mining. The importance of frequent pattern mining can be seen in different applications including bioinformatics, marketing and social media and many other applications.

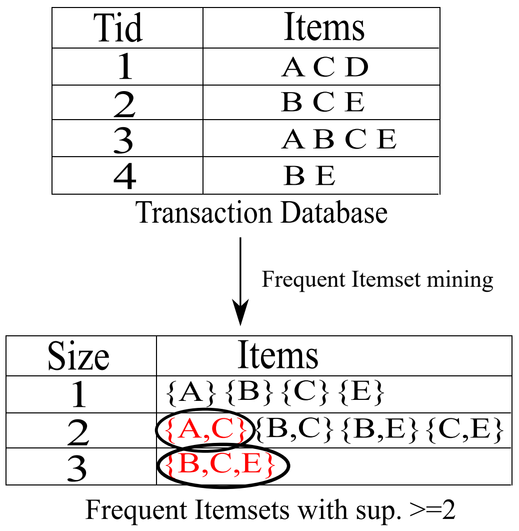
In this section, we present some of the important topics in frequent pattern mining. We will explain frequent itemset mining, frequent pattern mining in graphs and some related concepts and finally we explain the MULE algorithm which we extended in our work.

2.2.1. Frequent Itemset Mining

Let I be the set of all elements called items, $X = \{i_1, i_2, ..., i_k\} \subseteq I$ is called an itemset with k elements. A transaction is a tuple $T = (tid, X)$ where *tid* is a unique transaction identifier and X is an itemset. A set of transactions is called transactions database [6]. Figure 2.4 shows an example of a transactions database with four transactions and five unique items.

The support of an itemset X in a transactions database D is defined as the number of transactions in D that contains the itemset X and is denoted as $sup(X, D)$. An Itemset X is considered to be frequent in the transactions database D if $sup(X, D) \geq \delta$ where δ is the minimum support threshold defined by a user. For example, figure 2.4 in the second table presents frequent itemsets with size one, two, and three with minimum support threshold equals 2.

Frequent itemset mining can be defined as the process of finding the set of all frequent itemsets $F = \{X_1, X_2, ...\}$ where $sup(X_i, D) \geq \delta$ in the transactions database D [6].



Tid	Items
1	A C D
2	B C E
3	A B C E
4	B E

Transaction Database

Size	Items
1	{A} {B} {C} {E}
2	{A,C} {B,C} {B,E} {C,E}
3	{B,C,E}

Frequent Itemsets with sup. >=2

Figure 2.4. Frequent Itemset mining example with min support = 2.

Generating all itemsets and then finding the frequent ones among them is too expensive and usually the execution will not finish in a reasonable amount of time since it takes an exponential time to search the transaction database. Several pruning techniques and algorithms have been proposed to overcome this problem. **The monotonicity property** of the support of an itemset can be helpful in limiting the size of the search space. The support of an itemset is considered to be *monotone decreasing* when the itemset is being expanded to a larger itemset [6].
Formally, let X, Y be two itemsets where $X \subseteq Y$, then $supp(X) \leq supp(Y)$.

Using the monotonicity property, we can be sure that any superset of an itemset X can not be frequent if X is not frequent. This observation can help reduce the search space significantly. Another interesting observation from the monotonicity property is that any subset of a frequent itemset must be frequent which means we do not have to report all the itemsets, instead we can only report only the frequent itemsets that have no frequent supersets which are called **maximal frequent itemsets**. Figure 2.4 presents and example of using the monotonicity property to reduce the search space. After finding frequent itemsets with size 1, we notice that itemset {D} appears only one time, we can prune it because we know it can not have any frequent supersets since it is not frequent. Next step is to check every possible combination between the frequent itemsets of size 1 to create itemsets with size 2. The result is also a four 2-itemsets ({A,C},{B,C},{B,E},{C,E}). Next, we check frequent itemsets with size 3, the result is one 3-itemset ({B,C,E}).

A maximal frequent itemsets is a frequent itemset that does not have a frequent superset. Reporting only maximal frequent itemset will reduce the size of the output without any information loss, since any other frequent itemset must be as subset of these maximal itemsets. Note that the itemsets marked in red and circled in the second table in figure 2.4, ({A,C},{B,C,E}) do not have any frequent superset and all the other itemsets are subsets of these two itemsets which means that ({A,C},{B,C,E}) are maximal frequent itemsets.

2.2.2. Frequent Subgraph Mining

Since graphs are one of the most common ways to represent data, mining frequent subgraphs from a graph dataset is considered to one of the most important problems in data mining. The different data types that can be easily represented as graphs like text or images or videos or other scientific data like chemical compounds or genes expression profiles makes the frequent subgraph mining a very popular research problem [11].

Frequent subgraph mining can be defined as the process of discovering frequent subgraphs given a dataset of graphs. Frequent subgraph mining is similar to frequent itemset mining. A frequent subgraph is a subgraph with a support greater than or equal the user defined support threshold. The support of a subgraph is the number of graphs that contain that subgraph in the dataset.

The basic idea of most frequent subgraph mining algorithms is to start from a node or an edge then start growing the subgraph. Every time the subgraph is extended, the algorithm checks if the new subgraph is frequent or not [11]. Figure 2.5 presents an example of five graphs and shows an example of frequent pattern in these graphs with min support = 0.3. An easy way to find if the pattern is frequent or not is to associate with every unique edge in the graphs dataset an attribute vector to indicate in which graph the edge appeared. For example, the edges occurrence table shown in figure 2.5 shows the edges and the corresponding graphs in which they appeared. One can simply look at the common graphs between the edges to find the graphs where all the edges appeared together. In figure 2.5, the four edges appeared together in the first and the second graph which means the subgraph appeared in two out of five graphs with support = $2/5 = 0.4$ which makes this subgraph frequent since it has a support greater than the min support.

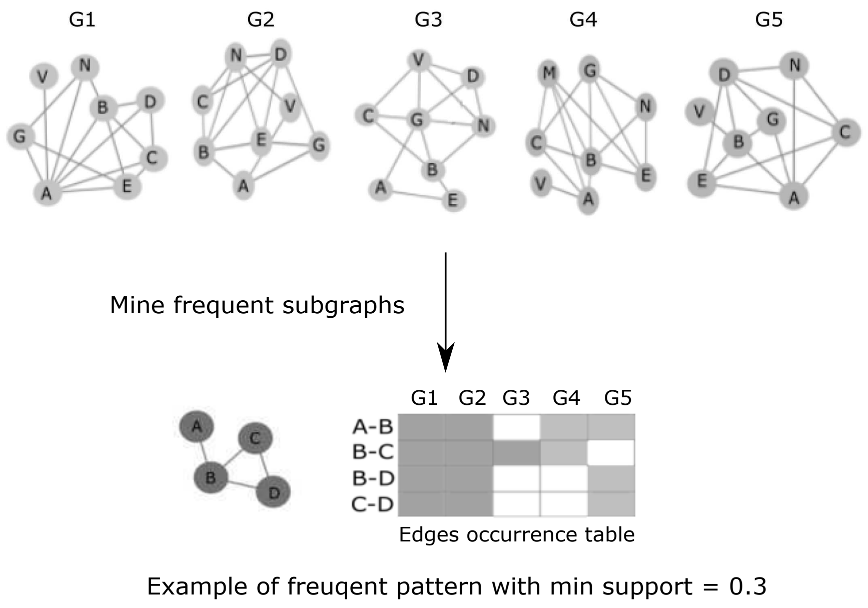


Figure 2.5. Frequent subgraph mining example.

Most of the techniques applied to the frequent itemset mining to prune the search space can be applied to frequent subgraph mining including the monotonicity property and the maximal frequent subgraphs. Generally, the two main factors that determine how efficient a frequent subgraphs mining algorithm are: First, the way the new subgraph is generated. Second, how to determine if the subgraph is frequent or not [11]. Many algorithms have been proposed to improve these factors and to prune the search space in an efficient way. One of the interesting algorithms for frequent subgraph mining is called **MULE** by Koyuturk et al [12]. We extended this algorithm in our work for discriminative subgraph mining and we will explain it in details in the next section.

2.2.3. MULE Algorithm

The MULE algorithm is an algorithm for detecting frequent subgraphs in biological networks algorithm proposed by Koyuturk et al [12]. The algorithm assumes the unique labeling of the nodes and edges of the graphs which simplify the problem significantly. It uses depth-first search enumeration starting from edges not nodes. The algorithm maintains four different sets for every pattern during the execution: The set of candidate edges, the set of visited edges, the set of edges that represent the current subgraph and the set of maximal frequent subgraphs.

The set of candidate edges is a set of edges that are directly connected to the current subgraph not visited yet. Enumerating only candidate edges minimize the search space and result in only connected subgraphs. The set of visited edges contains the edges that have been enumerated in the search regardless of whether they have been added to the subgraph or not. The set of edges that represent the current subgraphs contains the edges in the subgraph that being extended by the algorithm. Finally, the set of maximal frequent subgraphs contains the maximal frequent subgraphs which we are interested in.

The algorithm executes the following steps on every unique edge in the graphs dataset:

1. Pick an edge from the set of candidate edges which initially contains the neighbors of the starting edge.

2. Mark the current edge as visited by adding it to the set of visited edges.

3. Add the current edge to the current subgraph and check if it is still frequent.

4. If the new subgraph is frequent, update the candidate set by adding the incident edges of the new edge and run the algorithm on the new subgraph.

5. If the subgraphs can not be extended anymore, check if it has no superset in the set of maximal frequent subgraphs. If so, add it to the set of maximal frequent subgraphs and return.

The MULE algorithm is used for mining frequent subgraphs, we can extend it to be used for mining discriminative subgraphs by checking for discriminative edges instead of frequent edges. Moreover, the performance can be significantly improved by extending the current subgraphs with the best discriminative edge from the set of candidate edges instead of extending all the edges.

2.3. Genes Coexpression Networks

Genes Coexpression networks are networks in which nodes represent genes and there is an edge between two nodes if the two gene expression profiles are similar [25]. There have been numerous studies that show a relation between genes coexpression networks and the genes functionality. Genes with similar expression profiles most likely have similar functionality [25, 3]. This provides a cheap and efficient way to gain more information about the functionality of genes from other genes by using their genes coexpression networks. Figure 2.6 presents an example of genes expression profiles as a matrix and the resulting genes coexpression network created from that matrix.

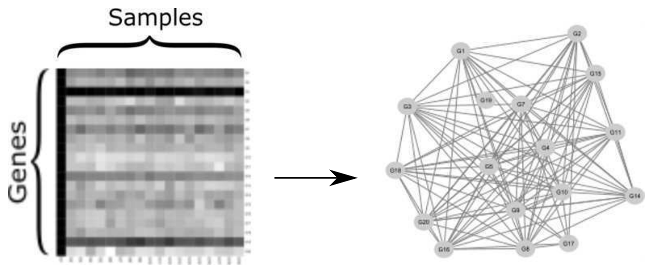


Figure 2.6. Gene coexpression network example.

Usually correlation is used to find the similarity between genes expression profiles across specific samples or tissues. If the correlation between two genes is above a specific threshold, then these two genes will be connected in the coexpression network. The most common way is to use the *Pearson correlation coefficient* as a co-expression measure to create the network [25].

After choosing the similarity measure, the next step is to create the similarity matrix. We denote the similarity matrix as a, an $n \times n$ matrix where n is the number of genes. The value at cell a_{ij} in the matrix represents how similar the i^{th} and j^{th} genes are. If Pearson correlation is used as the similarity measure, the value of a_{ij} equals the correlation between the i^{th} and j^{th} genes.

Genes coexpression networks can be weighted or unweighted. For a weighted network, the similarity matrix values will be the weight of the edges between the nodes. For an unweighted network, the similarity matrix must be converted to a binary adjacency matrix using a chosen threshold value. Any value in the similarity matrix greater than or equal to the threshold value will become 1 and any value less than the threshold will become 0. The resulting matrix is the adjacency matrix of the network.

Figure 2.7 presents an example of how coexpression networks are created. A gene expression profile matrix with size 20 × 19 with 20 genes and 19 samples, correlation matrix is created with size 20 × 20 that represents the correlation between the genes. Next, a cut off value is chosen to create an adjacency matrix. Any correlation value less than the cutoff will become zero, and any value greater than the cutoff will become 1. Using the adjacency matrix, any two genes with a value of 1 in the matrix will be connected in the coexpression network.

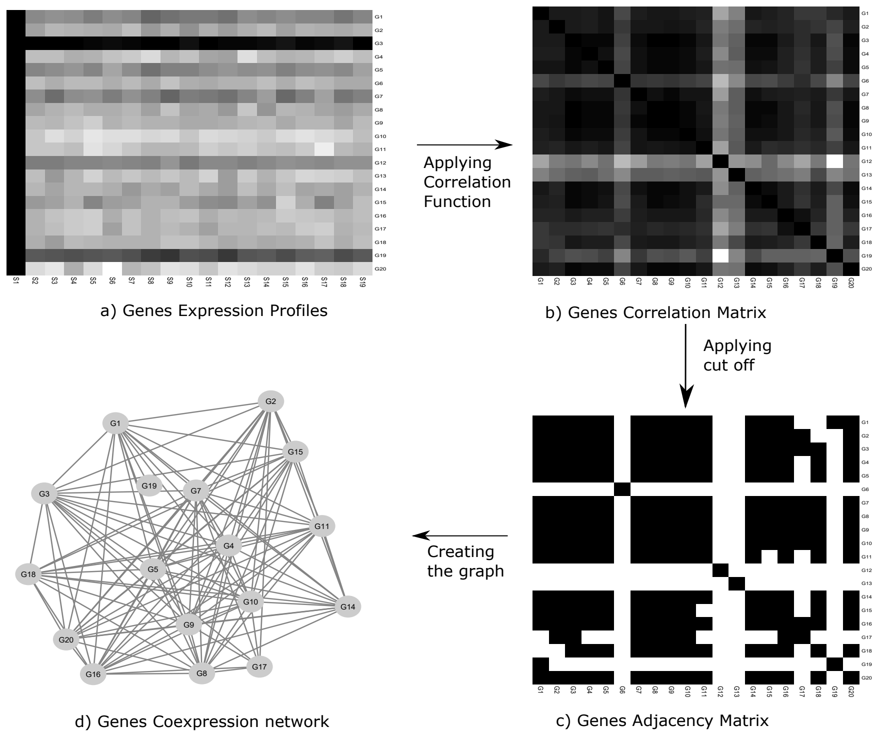

a) Genes Expression Profiles
Applying Correlation Function
b) Genes Correlation Matrix
Applying cut off
c) Genes Adjacency Matrix
Creating the graph
d) Genes Coexpression network

Figure 2.7. Coexpression network construction example.

2.3.1. Genes Coexpression Networks Classification

The problem of genes coexpression networks classification is to predict the class label of an unlabeled coexpression graph using a labeled coexpression graphs dataset. It is considered as one of the most popular techniques to gain new information about genes functionality and biological processes because its effectiveness and its reliable results which made it an attractive topic for many researchers.

Recent studies applied kernel methods in graph classification by creating a matrix that represents the pairwise similarity between graphs called kernel matrix and use that matrix for classification by applying a machine learning algorithm that used the matrix for classification of unlabeled graphs [22].

A different approach has been proposed by [13] that uses a topological features vector extracted from the graphs dataset for the classification. The idea is that similar graphs will have similar topological attributes. The authors showed that their approach is effective and comparable to the kernel methods. In another approach [21], the authors used **gSpan**, a frequent pattern mining framework, to find discriminative frequent subgraphs. These discriminative frequent patterns are then used for graph classification.

Finding all frequent subgraphs to use them for graph classification can be a daunting task. One must enumerate all frequent subgraphs for different values of a user specified threshold to find the threshold that produce the best results, not to mention the redundant work that have to be done since many of the frequent subgraphs will not be significant. An interesting technique is proposed in [23] to overcome this problem. The authors presented an algorithm called **LEAP** where they used structural proximity and frequency association to improve the scalability of the mining process.

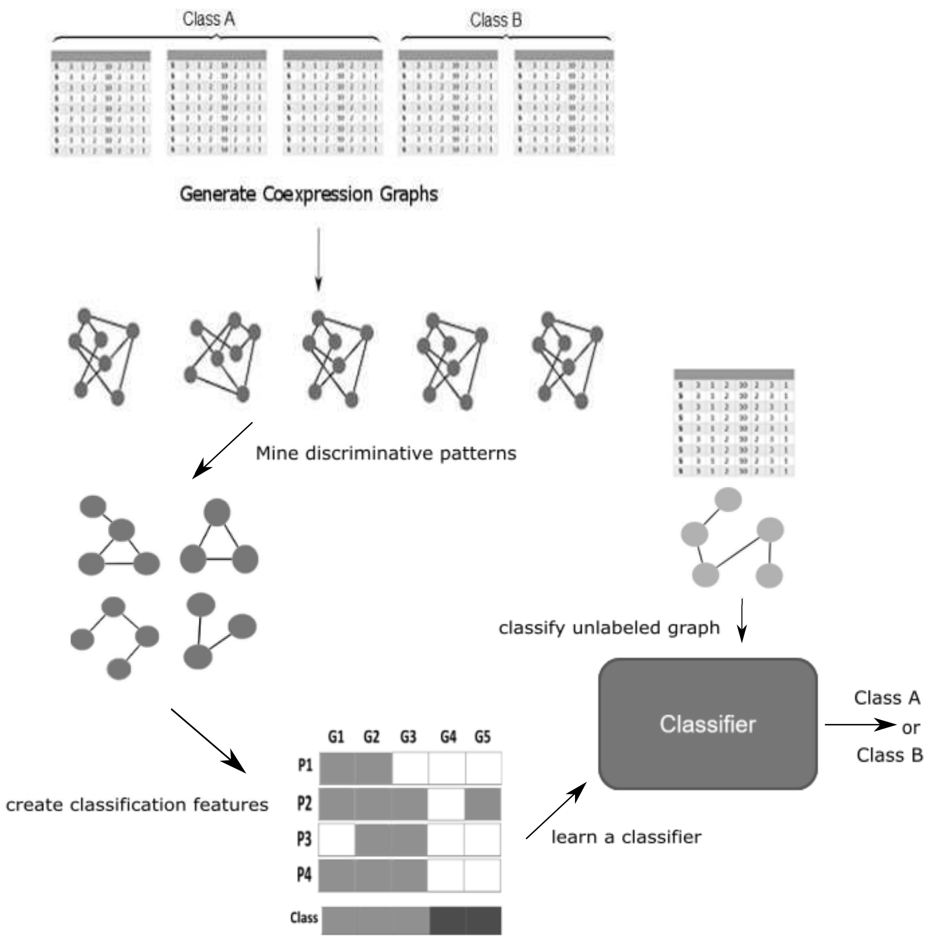


Figure 2.8. Gene coexpression networks classification algorithm.

In this work, we present two algorithms for genes coexpression networks classification using discriminative subgraphs mined from these networks. Figure 2.8 shows the process we follow in our algorithm to classify the genes coexpression networks. The process starts from discriminative patterns mining then moves to training the classifier using features extracted from the discriminative patterns and finally predicts the class label of unlabeled genes coexpression network. We propose two different objective functions to measure discriminative power of a subgraph and then we analyze the reported results. Moreover, we compare our two proposed algorithms to two baseline algorithms and show that our results outperform the baseline algorithms.

3. FIRST ALGORITHM

3.1. Problem Description

In this section we present a description of the problem of graph classification and some important definitions that will be used in our proposed algorithm.

Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, ..., v_n\}$ is the set of vertices, $E \subseteq V \times V$ is the set of edges, the graphs dataset is defined as $D = \{G_i\}_{i=1}^n$ where n is the number of graphs.

Given a graphs dataset $D = \{G_i\}_{i=1}^n$ and class $C = \{c_i\}_{i=1}^n$ where c_i is the class label of graph G_i. G_i the problem of graph classification is to predict the class label of an unlabeled graph using a training set of labeled graphs from D.

Given the set of all graphs $D = \{G_i = (V_i, E_i)\}_{i=1}^n$ where n is the number of graphs. The set of all edges can be defined as $E(D) = \bigcup_{i=1}^n E_i$.

For an easy and more importantly efficient processing of the graphs dataset, graphs can be represented as binary vectors in terms of $E(D)$. If an edge e appears in the graph G_i then that i_{th} value in the vector equal to 1, otherwise it would be 0. The complete representation of the graphs is a matrix where the rows are all the edges in $E(D)$ and columns are all graphs in D. The binary vectors of edges occurrence in the set of graphs can be used as an attribute vector for the corresponding edge to evaluate if the edge is discriminative or not. Figure 3.1, The edge occurrence table illustrate this idea where five graphs are represented as a binary vectors in terms of all the edges that appear in the graphs.

As mentioned in the introduction that recent research has shown the effectiveness of discriminative subgraphs for the tasks of gene expression classification. Numerous functions have been proposed for evaluating subgraphs to be discriminative or not. Most of these methods consider a subgraph to be discriminative if the edges of the subgraph are frequent in the graphs that are annotated with one class and less frequent in the graph annotated with the background class. These methods usually require that all the edges of the subgraph must appear together in the graphs which can been similar to applying logical AND function between the binary vectors of the edges occurrence and then evaluating the resulting vector.

An interesting approach would be to apply logical **OR function** on the binary vectors where at least one edge in the subgraph appears in the original graphs. The OR function places a more relaxed restriction on the subgraphs which gives it the ability to explore more interesting interactions between the nodes in the graphs. In gene coexpression networks, using the OR function could help reveal interesting undiscovered relationships between genes since some functions in the cell requires at least one interaction to exist which can be represented by logical OR function.

Formally, we can define i_{th} value in the OR binary vector of subgraph p as

$$OR(G', G_i) = \begin{cases} 1 & \exists \text{ e} \in E(G') \text{ and e} \in E(G_i) \\ 0 & \text{otherwise} \end{cases}$$

Where $E(G')$ is the set of edges in subgraph G' and $E(G_i)$ is the set of edges in graph G_i. Similarly, we define the whole OR binary vector as:
$OR(G', D) = \{OR(G', G_1), OR(G', G_2), \cdots, OR(G', G_n)\} = \{OR(G', G_i)\}_{i=1}^{n}$ where n is the number of graphs in D.

The AND function is more intuitive and it is what usually is used for mining subgraphs where all the edges must appear together in G_i to have a 1 in the i_{th} value in the AND binary vector of the subgraph G'.

For clear understanding of how these two functions are applied, Figure 3.1 presents an example where the two functions are used to mine discriminative subgraphs. The results of the AND function or OR function is used as an attribute vector to determine whether this is a good discriminative subgraph or not.

3.2. Algorithm

The proposed algorithm for graph classification investigates two major parts that highly affect the performance of the algorithm and quality of the subgraphs used for classification. First, the enumerating algorithm used to mine the discriminative subgraphs. Second, the objective function we used to decide if the subgraphs are discriminative or not and the attribute vector of the subgraphs we used as an input to the objective function.

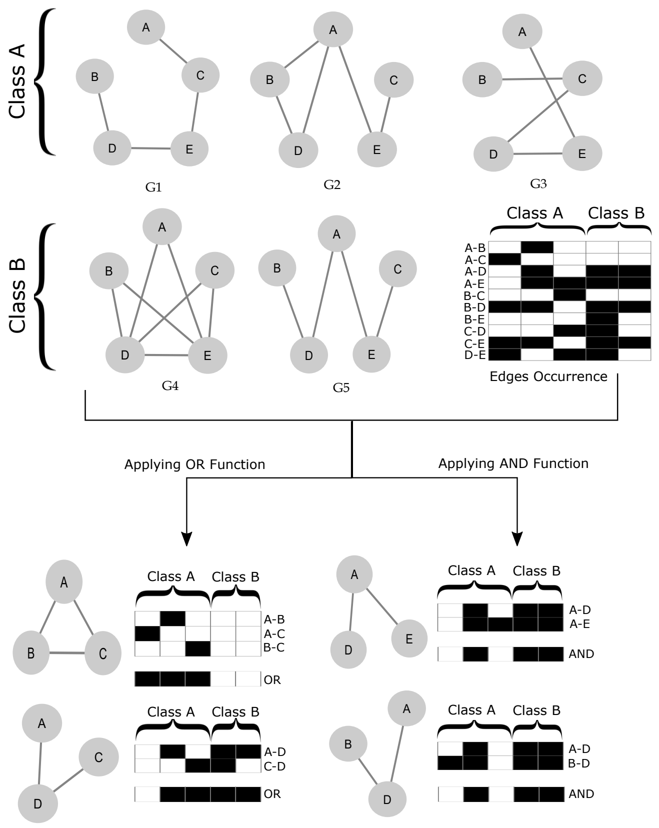


Figure 3.1. Example of applying AND and OR functions on edge occurrences.

3.2.1. Enumerating algorithm

As mentioned in the related work section, the algorithm we used to enumerate the discriminative subgraphs is a greedy-based algorithm. In our algorithm we will check for the discriminative edges that meet a user-specified minimum threshold.

The algorithm executes the following steps on every unique edge in the graphs dataset:

1. Pick an edge from the set of candidate edges which initially contains the neighbors of the starting edge.

2. Mark the current edge as visited by adding it to the set of visited edges.

3. Add the current edge to the current subgraph and check if it is still discriminative.

4. If the new subgraph is discriminative, update the candidate set by adding the incident edges of the new edge and run the algorithm on the new subgraph.

We extend the current subgraph with the best discriminative edge from the set of candidate edges.

3.2.2. Objective function

The objective function we used to evaluate the subgraphs whether discriminative or not is fairly simple. the objective function can be simply defined as:
$F(G) = \frac{R(G,D_A)}{|C_A|} - \frac{R(G,D_B)}{|C_B|}$ where $R(G, D_A)$ is the number of graphs annotated with class A and in which the subgraph G appears. In another words, the number of ones in the OR binary vector or AND binary vector of subgraphs G annotated with class A. The value of $F(G)$ should be greater than a user specified value δ to consider the subgraph G a discriminative subgraph. Note that the OR or AND binary vector of the subgraph can be obtained by simply applying a logical OR or AND on the binary edges occurrence vectors of edges in the subgraph.

The algorithm is executed on every edge in the $E(D)$ provided that the edge has a value of $F(G)$ that is greater that the user specified δ (see lines 2-6). Next, the algorithm starts adding edges from the candidate set and checking how the value of the discriminative function changes (see lines 9-18). The candidate set is the set of edges that are directly connected to current subgraph and has not been visited yet which is initially the set of directly connected edges $N(e)$ to current edge e . Following that, it chooses the edge that maximizes $F(G')$ as long as it is greater than the current $F(G)$ with a user specified improvement rate t. Since some edges might not add a significant improvement to the current subgraph, the use of t can be very helpful in pruning unnecessary extensions. Note that the candidate set is updated after extending the subgraph with the new edge and the algorithm is recursively called with the new subgraph (see lines 19-23). If the edge is not extended with any of the edges from the candidate set Cs, which means there are no new edges can be added to improve $F(G)$,the Boolean variable *extensible* will be set to false and the current subgraph will be added to the set S as a discriminative subgraph.

Algorithm 1 MineDS: Mining Discriminative Subgraphs for graph classification

Input:
Cs: set of non visited edges directly connected to current subgraph
Vs: set of visited edges
δ: minimum value of objective function of the subgraph
t: minimum improvement rate of objective function
Output:
$\mathcal{S}$: the set of non-extensible discriminative subgraphs

```
 1: $\mathcal{S} \leftarrow \{\}$
 2: for $e \in E(D)$ do
 3:     if $F(e) \geq \delta$ then
 4:         MineDS({e},$N(e)$,{})
 5:     end if
 6: end for
 7: function MineDS($G$,$Cs$,$Vs$)
 8:     $extensible \leftarrow false$
 9:     $max\_\delta \leftarrow 0$
10:     $max\_e \leftarrow null$
11:     for $c \in Cs$ do
12:         $Vs \leftarrow Vs \cup c$
13:         $G' \leftarrow G \cup c$
14:         if $F(G') \geq max\_\delta$ then
15:             $max\_\delta \leftarrow F(G')$
16:             $max\_e \leftarrow c$
17:         end if
18:     end for
19:     if $max\_\delta \geq F(G) + F(G) * t$ then
20:         $extensible \leftarrow true$
21:         $G' \leftarrow G \cup max\_e$
22:         $Cs \leftarrow Cs \cup N(max\_e) \backslash Vs$
23:         MineDS($G'$,$Cs$,$Vs$)
24:     end if
25:     if !$extensible$ then
26:         if $GhasnosupersetinS$ then
27:             $\mathcal{S} \leftarrow \mathcal{S} \cup G$
28:         end if
29:     end if
30: end function
31: return $\mathcal{S}$
```

3.3. EXPERIMENTS

In this section, we describe the gene expression datasets we used to evaluate our algorithm. We compare our algorithm with the MOSA algorithm proposed in [15] and to the algorithm proposed in [13]. Following that, we do an in-depth analysis for the patterns reported by our algorithm.

3.3.1. Dataset

The gene expression datasets used in the experiments are selected from the 338 GEO (**G**ene **E**xpression **O**mnibus) datasets used in [16]. We selected a 96 GEO datasets from the 338 GEO datasets and generated the 96 coexpression graphs. Our dataset has 2,760,546 edges and 13,838 nodes. For the class labels, the original dataset have a list of phenotypes and for each phenotype which datasets that are annotated with this phenotype. We ordered the phenotypes according to the number of datasets annotated by the phenotype and chose the top four phenotypes. These phenotypes are *"Leukocytes", "Neoplasms, Glandular and Epithelial", "Leukocytes, Mononuclear" and "Carcinoma"*. We applied our algorithm for each of these phenotypes as a class label and compared our results to the previously mentioned algorithms.

3.3.2. Topological Analysis of Discriminative Patterns

We ran our algorithm on each of the four phenotypes and analyzed the subgraphs or patterns reported by our algorithm. Table 3.1 presents our results using different values for δ, as for the improvement rate t we used 5% as improvement rate in all of our experiment. The table shows that most of the patterns are quite dense with density around 60% and 80% in most cases and average number of nodes $\overline{N}$ is 4 or greater than 4 in all of the cases except for the last phenotype with δ of 0.5 .

3.3.3. Graph Classification Accuracy

After analyzing the results reported by our algorithm, we compared our algorithm to two algorithms in terms of accuracy of the graphs classification. The first one is the MOSA algorithm proposed by [15] that used simulated annealing objective function to choose the next state and decides to add or remove a node from the current subgraph. The second one is the algorithm proposed in [13], it uses topological attributes extracted from the graphs dataset as features for the graph classification.

Table 3.1. Topological Analysis of the patterns reported by the first algorithm

Phenotype	δ	No. of patterns	$\overline{N}$	$\overline{Density}$
Leukocytes	0.2	5899	6	0.6
	0.3	1013	5	0.64
	0.4	244	5	0.61
	0.5	28	5	0.52
Neoplasms, Glandular and Epithelial	0.2	3887	5	0.74
	0.3	930	4	0.81
	0.4	350	4	0.82
	0.5	170	3	0.85
Leukocytes, Mononuclear	0.2	3029	5	0.62
	0.3	536	5	0.64
	0.4	107	5	0.59
	0.5	7	5	0.48
Carcinoma	0.2	12397	5	0.56
	0.3	1267	4	0.73
	0.4	465	4	0.8
	0.5	208	3	0.84

We used the OR binary vectors and AND binary vectors of the subgraphs reported by our algorithm as feature vector for the classification. Using the Weka data mining software [7], we ran the decision tree classification algorithm on the results reported by our algorithm using the OR binary vectors and then on the results using the AND binary vectors. Following that, we ran the same classification algorithm on the MOSA algorithm results and topological attributes algorithm results. The complete results presented in Table 3.2.

The classification accuracy using the patterns reported by our algorithm outperform both MOSA and topological attributes algorithm in all of the phenotypes. We can see that the MOSA results are better than the topological attributes results but in comparison to using AND or OR function, both functions perform much better. The difference between using AND or OR function when mining subgraphs as we explained before is that in AND, the whole subgraph appears in all of the graphs corresponding to the 1's in the binary attribute vector of the subgraph. On the other hand, OR function means at least one edge in the subgraph appears in the graphs corresponding to the 1's in the binary attribute vector of the subgraph. Both AND and OR perform very well in comparison to MOSA and topological attributes but comparing using AND function to using our proposed OR function algorithm, it is clear that OR is a much better choice.

Table 3.2. Classification accuracy using OR, AND, Topology and MOSA algorithms.

Phenotype	OR	AND	Topology	MOSA
Leukocytes	95.83%	93.75%	85.41%	85.41%
Neoplasms, Glandular and Epithelial	93.75%	88.54%	78.12%	87.5%
Leukocytes, Mononuclear	94.79%	91.67%	78.12%	87.5%
Carcinoma	97.91%	89.58 %	89.58%	89.58%

A closer look can be provided by creating the heatmap for the OR binary vectors of the top patterns reported by our algorithm. We ordered the reported patterns according to their discriminative function value and then chose the top 70 patterns and created the heatmap for them. Figures 3.2, 3.3 , 3.4, and 3.5 present the results for the four phenotypes with a value of $\delta = 0.4$

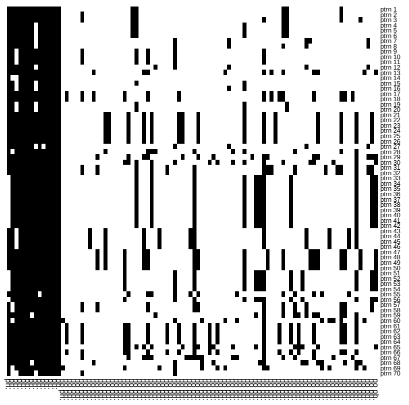


Figure 3.2. Heatmap for top 70 patterns (phenotype "Leukocytes" and δ= 0.4)

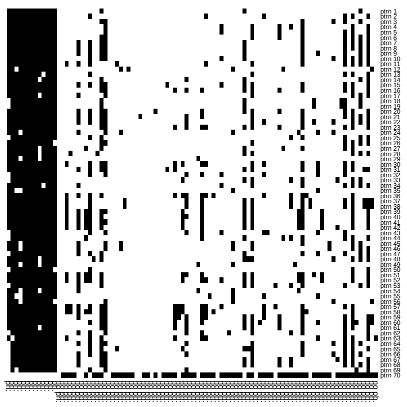


Figure 3.3. Heatmap for top 70 patterns (phenotype "Neoplasms, Glandular and Epithelial" and δ= 0.4)

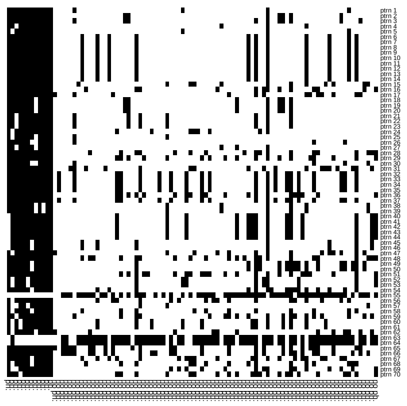


Figure 3.4. Heatmap for top 70 patterns (phenotype "Leukocytes, Mononuclear" and δ= 0.4)

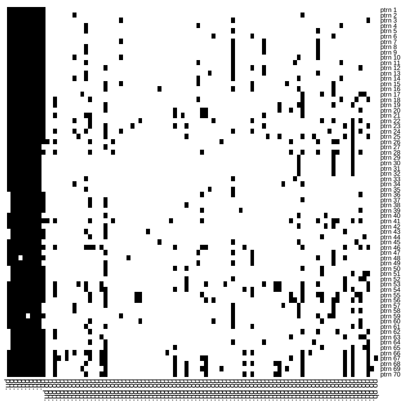


Figure 3.5. Heatmap for top 70 patterns (phenotype "Carcinoma and δ= 0.4).

The black cells in the figures indicate a value of 1 and the white cells indicates a value of zero. Each row represent a pattern and each column represents a graph. It is clear from the heatmap that the columns of the graphs annotated with the phenotype are mostly black cells and the columns of graphs annotated with the background class are mostly white cells. This clearly shows the ability of the algorithm in detecting and reporting patterns that can be used for graph classification with a very high accuracy.

3.3.4. Enrichment Anaylsis

Biological enrichment analysis was performed on the patterns reported by our algorithm to gain more information about the biological significance of the reported patterns. For the enrichment analysis, we used **D**atabase for **A**nnotation, **V**isualization, and **I**ntegrated **D**iscovery - DAVID [8, 9]. We also used **clusterProfiler** a library in R programming language that is used for analyzing and visualizing functional profiles (GO and KEGG) of gene and gene clusters [24]. We used DAVID for GO term and KEGG enrichment analysis and used **clusterProfiler** library for disease enrichment analysis in the reported patterns.

Figure 3.6 presents our results for the percentage of enriched patterns in GO terms for different values of δ. From Figure 3.6 we can see that the percentage of the enriched patterns ranges from about 45% to more than 60% for the four different phenotypes at a value of $\delta = 0.2, 0.3, and 0.4$. The percentage of enriched patterns decreases when we increase the value of δ since the number of patterns decreases when δ increases.

Figure 3.7 presents the results of the KEGG enrichment analysis. By looking at the results, we can see the percentage of the enriched patterns at a low value of $\delta = 0.2$ is between 20% and 30%, then when we increase δ to 0.3 the enrichment improves then the percentage decreases as we increase δ. This observation can help us determine the best value for δ and as we can see from Figure 3.6 and Figure 3.7, the best value for δ is between 0.3 and 0.4.

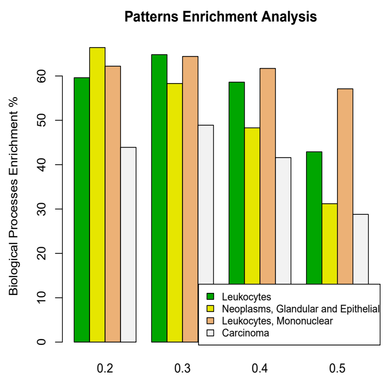


Figure 3.6. Patterns Biological Processes Enrichment Analysis.

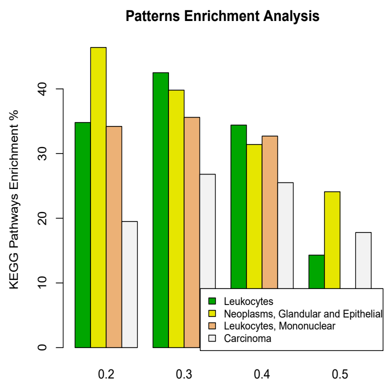


Figure 3.7. Patterns KEGG Pathways Enrichment Analysis.

We also analyzed the top GO terms in the reported patterns for every phenotypes and created a heatmap of the result. Figures 3.8, 3.9, 3.10, and 3.11 present the top 30 GO terms significantly enriched in the patterns reported by our algorithm. The rows represent the GO terms and the columns are the patterns. A black cell indicates that the GO term is enriched in the corresponding pattern. The GO biological processes terms include *selenium compound metabolic process*, *SRP-dependent cotranslational protein targeting to membrane*, *protein targeting to ER*, *positive regulation of lymphocyte activation*, *B cell mediated immunity*, *protein localization to endoplasmic reticulum*, and *extracellular matrix disassembly*.

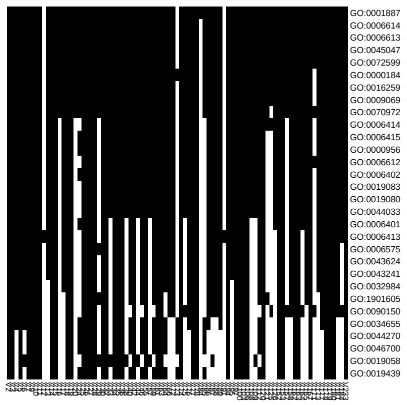


Figure 3.8. Heatmap for top 30 GO terms enriched in the phenotype (Leukocytes).

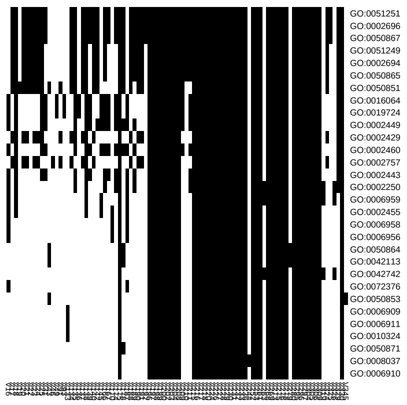


Figure 3.9. Heatmap for top 30 GO terms enriched in the phenotype (Neoplasms, Glandular and Epithelial).

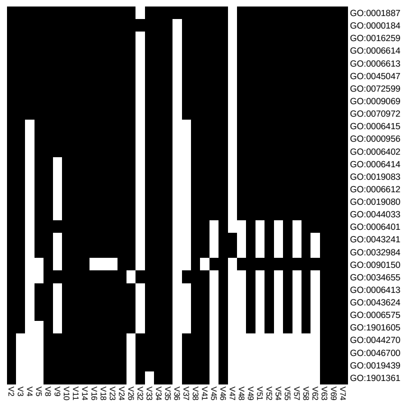


Figure 3.10. Heatmap for top 30 GO terms enriched in the phenotype (Leukocytes, Mononuclear).

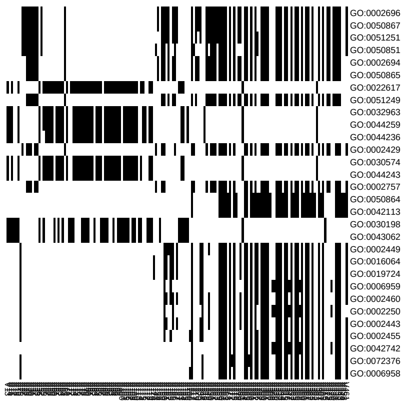


Figure 3.11. Heatmap for top 30 GO terms enriched in the phenotype (Carcinoma).

Moreover, we checked the disease enrichment for the patterns and reported the top 5 diseases enriched in the reported patterns for every phenotype. The results are summarized in Table 3.3. Other diseases enriched in the patterns include *gingival disease*, *chronic leukemia*, *abdominal aortic aneurysm*, *breast ductal carcinoma*, *osteochondrodysplasia*, and *reproductive organ benign neoplasm.*

Table 3.3. Top 5 diseases enriched in the first algorithm's phenotypes patterns.

Phenotype	Top Diseases
Leukocytes	• mouth disease • Periodontal disease • Tooth disease • Bacterial vaginosis • Periodontitis
Neoplasms, Glandular and Epithelial	• hypersensitivity reaction type IV disease • Sarcoidosis • Collagen disease • Lupus erythematosus • Ehlers-Danlos syndrome
Leukocytes, Mononuclear	• prostate adenocarcinoma • Autism spectrum disorder • Autistic disorder • Pervasive developmental disorder • Prostate carcinoma
Carcinoma	• collagen disease • Ehlers-Danlos syndrome • Hypersensitivity reaction type IV disease • Sarcoidosis • Uterine benign neoplasm

4. SECOND ALGORITHM

4.1. Problem Description

In this section we present some important definitions that will be used in our second proposed algorithm and some important notes about our second algorithm.
Let $G = (V, E)$ be an undirected graph, where $V = \{v_1, ..., v_n\}$ is the set of vertices, $E \subseteq V \times V$ is the set of edges, the graphs dataset is defined as $D = \{G_i\}_{i=1}^{n}$ where n is the number of graphs.

Given a the graphs dataset $D = \{G_i\}_{i=1}^{n}$ and class $C = \{c_i\}_{i=1}^{n}$ where c_i is the class label of graph G_i, The problem of graph classification is to predict the class label of an unlabeled graph using a training set of labeled graphs from D.

Given the set of all graphs $D = \{G_i = (V_i, E_i)\}_{i=1}^{n}$ where n is the number of graphs. The set of all edges can be defined as $E(D) = \bigcup_{i=1}^{n} E_i$.

As we mentioned in the first algorithm, graphs can be represented as binary vectors in terms of edges occurrences and D. If an edge e appears in the graph G_i then that i_{th} value in the vector equal to 1, otherwise it would be 0. The complete representation of the graphs is a matrix where the rows are all the edges in $E(D)$ and columns are the all graphs in D. The rows in that matrix represent the occurrences of the corresponding edges, these binary vectors of edges occurrence can be used as an attribute vector for the corresponding edge to evaluate if the edge is discriminative or not. In figure 4.1, the edge occurrence table illustrate above technique where five graphs are represented as a binary vectors in terms of all the edges that appears in the graphs.

A traditional approach that uses the edges occurrences for mining discriminative patterns is to find patterns with edges that are frequent in graphs labeled with the positive class and not frequent in graphs labeled with the negative class. A pattern or subgraph has a binary attribute vector that shows where all the edges of the subgraph appeared together in the graphs dataset. The pattern attributes vector is used to evaluate how discriminative that patterns by finding the difference between the relative support of the pattern in the positive class and the relative support of the patterns in the negative class. The relative support of a pattern in class i can be defined as the number of attributes with a value of 1 labeled with class i over the total number of attributes labeled with class i in the attribute vector of the pattern.

Unlike the frequency measure, the discriminative measure is not antimonotone. This means that the search can not be pruned once we reach a pattern that does not meet the minimum specified threshold. An Interesting antimonotonic measure for discriminative power of an itemset proposed by [4] called *SupMaxK* which is calculated by finding the difference of a itemset support in one class and and the maximal support between all of its size-K subsets of the itemset in the other class. The smaller value of k the more effective the approach in finding low-support discriminative patterns. What makes this algorithm effective is that it is antimonotonic which means any superset of a non-discriminative pattern discovered by the mining algorithm can not be discriminative and can be pruned.

The authors in [4] formally define *SupMaxK* of an itemset as follow:

$$SupMaxK(\alpha) = RelSup^{+}(\alpha) - \max_{\beta \subseteq \alpha}(RelSup^{-}(\beta))$$
$$where|\beta| = K$$

In our algorithm, we extended this discriminative measure *SupMax2* to mine connected subgraphs instead of itemsets. We chose $K = 2$ to get the best possible results by our proposed algorithm.

4.2. Algorithm

In this section we present our second proposed algorithm for graph classification. The main task is similar to our first proposed algorithm where we mine the graphs dataset for discriminative patterns and use these patterns as features for classification on an unlabeled graph. In the second algorithm, we followed different approach to mining the discriminative patterns and presented some pruning techniques to minimize the search space. We will present first the enumerating algorithm used to mine the discriminative subgraphs. Then, we provide more details about how we used the discriminative measure $SupMax2$ and explain the pruning strategies we utilized to make the search faster and more efficient. Lastly, we present the full algorithm pseudo code.

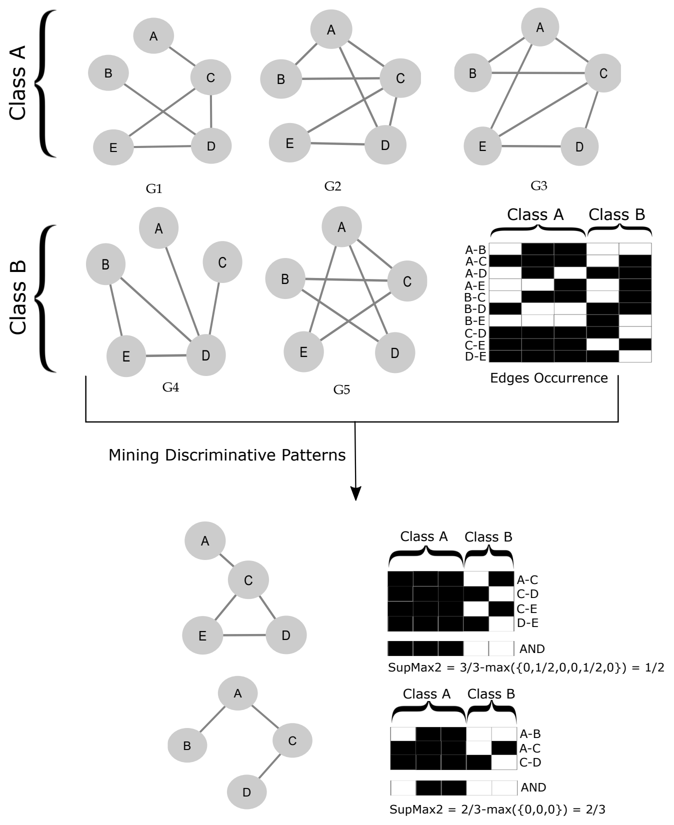


Figure 4.1. Example of SupMax2 function on edge occurrences table.

4.2.1. Enumerating algorithm

The enumerating MULE algorithm proposed by Koyuturk et al[12] which we extended in the first algorithm will be used here with few changes. The algorithm will be used to enumerate every possible subgraph instead of following a greedy approach as we did in the first algorithm. Since this approach will be much slower and number of reported patterns will be huge, we utilized few pruning strategies that we will mention later in the following subsections.

The algorithm starts from discriminative edges and grows by adding new edges to the subgraph as long as the new subgraph meets the minimum discriminative threshold. The algorithm stops when adding new edges results in subgraph that does not satisfy the discriminative threshold condition.

4.2.2. Discriminative Measure

The discriminative power of a pattern is usually measured by calculating the difference between the relative support of the pattern in positive class and the relative support of the pattern in the negative class. The problem with this measure is that it is not antimonotonic, a pattern that is not discriminative might become discriminative if new edge is added which means it can not be pruned. Such problem requires to enumerate all subgraph of the graph to find the discriminative patterns.

As mention earlier, the authors in [4] proposed a discriminative measure called $SupMaxK$ that is antimonotonic, meaning that if we discover a pattern that is not discriminative according to the $SupMaxk$ measure, we can prune the search tree. This makes the search much faster and efficient.

For our algorithm, we extended the $SupMaxK$ to mine connected discriminative subgraphs and we chose the $k = 2$ to be used in our algorithm. To calculate the $SupMax2$ of a pattern, we need to calculate the maximum support value between all the size-2 subsets of the pattern in the negative class and subtract that value from the pattern relative support in the positive class. If the value of $SupMax2$ is greater than a user specified threshold, then that pattern is considered discriminative.

Figure 4.1 present an example of how our algorithm works on the graphs dataset. Using the 5 graphs as a graph dataset where the first 3 graphs are labeled as class A or positive class and the last 2 graphs are labeled with class B or negative class, we mine for discriminative patterns that satisfy the *SupMax2* minimum threshold value. An example of two discriminative patterns using a $SupMax2 \geq 0.5$ is presented in the figure. The first pattern have a $RelSup^{+} = 3/3 = 1$, and the maximal support of all size-2 subsets in the negative class is $max(0, 1/2, 0, 0, 1/2, 0) = 1/2$. $SupMax2 = 3/3 - 1/2 = 1/2$ which is equal to the minimum threshold value and the pattern is discriminative.

4.2.3. Pruning strategies

Mining discriminative patterns from a large graph dataset can take a very long time even when using an antimonotonic measure. For this reason, we applied pruning strategies to make the search more efficient and reduce the size of the search tree.

First thing we did to reduce the running time, we calculated the $SupMax2$ between every two edges and stored it in a matrix so that we do not have to calculate it again when running the algorithm.

1. **Pruning based on the positive relative support**

   Lemma 1. *Given the set of all edges $E(D)$, any edge e_i in $E(D)$ when extended with another edge e_j will have a $SupMax2$ value less that the relative support of any of the edges in the positive class, $RelSup^+(e_i)$ and $RelSup^+(e_j)$. Thus, $RelSup^+(e)$ is an upper bound for any pattern that have edge e in it.*

   Since the $RelSup^+$ in a non increasing function and the maximal support between all the size-2 subsets of the pattern in the negative class is a non decreasing function, we can guarantee that the $SupMax2$ value will always be less than $RelSup^+$ for all the edges in a pattern. Using this lemma, we can prune all edge with $RelSup^+$ less than the minimum discriminative threshold from the beginning and only call the algorithm on the edges that satisfy the minimum threshold constraint. Moreover, we can apply the same pruning in the algorithm when a new pattern is created. We can check the $RelSup^+$ of the new pattern and only do the recursive call on the patterns that satisfy the min threshold constraint.

2. **Pruning based on old vale of** $max(RelSup^-)$

   Lemma 2. *Given a pattern α and a $max(RelSup^-(\beta))$ where β is any size-2 subset of the pattern α, A new pattern α' created by extending α with edge e cannot have a $SupMax2$ value that is greater than $UB(\alpha') = RelSup^+\alpha'$ - $max(RelSup^-(\beta))$. $UB(\alpha')$ can be used as an upper bound on the pattern α'*

   Since $max(RelSup^-)$ is non decreasing function, the value for the pattern α will always be less than or equal the value for the pattern $\alpha' = \{e\} \cup \alpha$. This upper bound can be very helpful, especially when the number of good edges at the start of the search is too large that we can not pre-calculate the $SupMax2$ for every edge and store the values in a matrix as we mentioned before. Using this upper bound reduces the number of times we need to calculate $SupMax2$.

3. **Pruning based on drop rate of the** $SupMax2$ **value**

   Sometimes when a pattern is extended by one edge, the value of the $SupMax2$ drops significantly but remains above the min threshold. For example, If we have min threshold

$\delta = 0.5$, a pattern with 5 edges and $SupMax2 = 0.85$ could become a pattern with 6 edges and $SupMax2 = 0.55$ but remains above the value of δ. Clearly, the pattern with $SupMax2 = 0.85$ is much better. As a solution for this problem, we used a drop rate threshold so that if the value of $SupMax2$ drops significantly with drop rate less than a user specified value dr, we do not extend the pattern even if the new value of $SupMax2$ is still greater than the value of δ.

Definition 1. *Given a pattern P and a pattern $P' = P \cup \{e\}$, the drop rate of P' is defined as $DRate(P') = \frac{SupMax2(P) - SupMax2(P')}{SupMax2(P)}$*

For a pattern to get extended, its drop rate must be less than the user specified threshold dr.

Lemma 3. *Given a pattern P and a pattern $P' = P \cup \{e\}$ and $UB(P')$ is the upper bound on the value of $SupMax2(P')$, the value of $DRate(P')$ cannot be greater than $UB_{DropRate}(P') = \frac{SupMax2(P) - UB(P')}{SupMax2(P)}$. $UB_{DropRate}(P)$ is an upper bound on the drop rate of pattern P.*

This upper bound is also very helpful when the number of good edges at the start of the search is too large that we can not pre-calculate the $SupMax2$ for every edge and store the values in a matrix.

The algorithm is executed on every edge in the $E(D)$ provided that the edge has a value of $RelSup^{+}$ that is greater that the user specified δ (see line 2-6). Next, when the algorithm is called on the edge, it starts adding edges from the candidate set to the visited set and checks if the new pattern can be pruned using the previously mentioned pruning strategies. If the new pattern satisfies all the conditions, maximal flag is set to false and the candidate set is updated and the algorithm is executed on the new pattern. Otherwise, that branch will be pruned and the pattern will be added to the set of the maximal patterns if it has no superset in that set.

Algorithm 2 MineMaxDisc: Mining Discriminative Subgraphs Using SupMax

Input:
Cs: set of non visited edges directly connected to current subgraph
Vs: set of visited edges
δ: minimum value of discriminative score
dr: minimum drop rate of discriminative score
Output:
$\mathcal{S}$: the set of maximal discriminative subgraphs

```
 1: $\mathcal{S} \leftarrow \{\}$
 2: for $e \in E(D)$ do
 3:     if $RelSup^{+}(\{e\}) \geq \delta$ then
 4:         MineMaxDisc({e},$N(e)$,{})
 5:     end if
 6: end for
 7: function MineMaxDisc($P$,$Cs$,$Vs$)
 8:     $maximal \leftarrow true$
 9:     for $c \in Cs$ do
10:         $Vs \leftarrow Vs \cup c$
11:         $P' \leftarrow P \cup c$
12:         if $RelSup^{+}(P') \geq \delta$ then
13:             if $|P| > 1$ then
14:                 if $UB(P') > 0$ & $UB_{DropRate}(P') \leq dr$ then
15:                     if $SupMax2(P') \geq \delta$ & $DRate(P') \leq dr$ then
16:                         $maximal \leftarrow false$
17:                         $Cs \leftarrow Cs \cup N(c) \backslash Vs$
18:                         MineMaxDisc($P'$,$Cs$,$Vs$)
19:                     end if
20:                 end if
21:             else
22:                 if $SupMax2(P') \geq \delta$ & $DRate(P') \leq dr$ then
23:                     $maximal \leftarrow false$
24:                     $Cs \leftarrow Cs \cup N(c) \backslash Vs$
25:                     MineMaxDisc($P'$,$Cs$,$Vs$)
26:                 end if
27:             end if
28:         end if
29:     end for
30:     if $maximal$ then
31:         if P has no superset in S then
32:             $\mathcal{S} \leftarrow \mathcal{S} \cup P$
33:         end if
34:     end if
35: end function
36: return $\mathcal{S}$
```

4.3. EXPERIMENTS

In this section, we describe the datasets used to evaluate our second algorithm. We compared our algorithm with the MOSA algorithm proposed in [15] and to the algorithm proposed in [13] that uses topological attributes for graph classification. Following that, we do an in-depth analysis for the modules reported by our algorithm.

4.3.1. Dataset

For our experiment, we selected 96 GEO gene expression datasets from the 338 GEO datasets used in [16]. We constructed the coexpression networks using these datasets. The co-expression networks have 2,760,546 edges(interactions) and 13,838 nodes(genes) .The set of all edges was too large to process so we decided to include only the edges that appear in five or more graphs since any edge that appears in less than five graphs is unlikely to be discriminative. After removing these edges the dataset contained 91,421 edges and 13,838 nodes. For the class labels, the original dataset have a list of phenotypes associated with 338 GEO dataset and for each phenotype which datasets that are annotated with this phenotype. We ordered the phenotypes according to the number of graphs annotated by the phenotype and chose three phenotypes as class labels for the graphs. These phenotypes are *"Leukocytes", "Leukocytes, Mononuclear" and "Neoplasms"*. We ran our algorithm for each of these phenotype as a class label and compared our results to the previously two previously mentioned algorithms (MOSA [16] and topological attributes-based algorithms [13]).

4.3.2. Topological Analysis of Discriminative Patterns

We ran our second algorithm on each of the three phenotypes and analyzed the patterns reported by our algorithm. Table 4.1 presents our results using different values for δ, as for the drop rate dr we chosen 20% as the allowed drop rate for any pattern for the whole experiment. We can see that most of the patterns are quite dense with density around 100% in most cases and average number of nodes, $\overline{N}$, is 4, 5 or 6 in all of the cases except for the last phenotype. As we can see that when the value of δ drops the number of patterns increases significantly. The smallest value of δ we could process and analyze its result is 0.4, for $\delta \leq 0.3$ the number of reported patterns is too large for processing.

Table 4.1. Topological Analysis of the patterns reported by the second algorithm

Phenotype	δ	No. of patterns	$\overline{N}$	$\overline{E}$
Leukocytes	0.4	299425	6	15
	0.5	824	5	10
Leukocytes, Mononuclear	0.4	25808	6	14
	0.5	98	4	5
Neoplasms	0.4	830	4	8
	0.5	25	3	3

4.3.3. Graph Classification Accuracy

After analyzing the results reported by our algorithm, we compared our second algorithm to two algorithms in terms of the accuracy of the graphs classification. First one is the MOSA algorithm proposed by [15] that uses simulated annealing function to choose the next state when mining for patterns and decides to add or remove a node from the current subgraph. The second one is algorithm proposed in [13], it uses topological attributes extracted from the graphs dataset as features for the graph classification.

We used the attributes vectors of the subgraphs reported by our algorithm as feature vector for the classification. The attribute vector is a binary vector that indicates in which graphs the subgraphs appeared. If there is a 1 in the i^{th} location in the vector, it means that the subgraphs appears in the i^{th} graph in the dataset. Using Weka data mining software [7], we ran the decision tree classification algorithm on the results reported by our algorithm and compared the results to the MOSA algorithm results and topological attributes algorithm results, the complete classification results of the three algorithms are presented in Table 4.2.

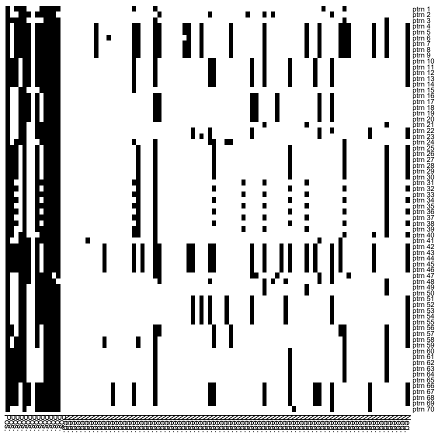
The classification accuracy using the patterns reported by our algorithm outperforms both MOSA and topological attributes algorithm in all of the phenotypes. We can see that the MOSA results are better than the topological attributes results but compared to our algorithm, our classification results are much better. We noticed that the last phenotype result is not very good which may be specific to this phenotype since the accuracy is low for all of the algorithms but even though the accuracy is lower than other phenotypes, our algorithms still outperforms the other two algorithms. We can see from the reported results that our algorithm can discover patterns that classify graphs with a very high accuracy.

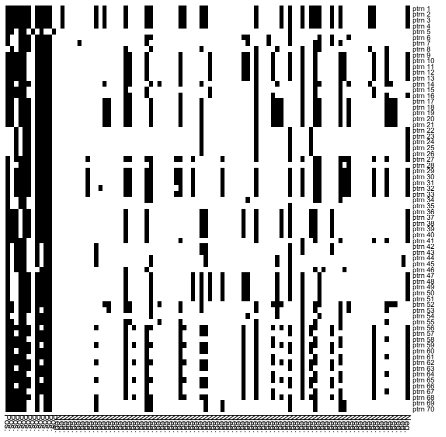
Table 4.2. Second Algorithm classification accuracy comparison.

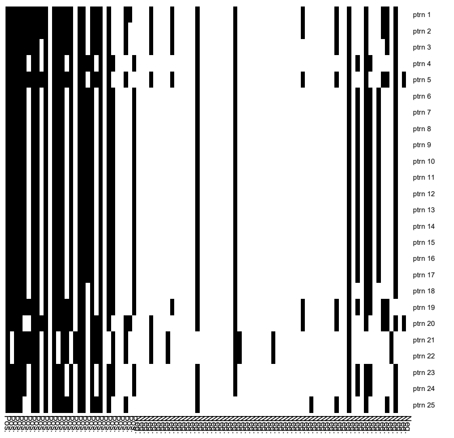
Phenotype	Proposed Algorithm	Topology	MOSA
Leukocytes	91.67%	86.45%	80.20%
Leukocytes, Mononuclear	96.87%	85.41%	85.41%
Neoplasms	76%	68.75%	68.75%

For an in-depth analysis of the reported results, we created a heatmap for the attribute vectors of the top patterns reported by our algorithm. We ordered the reported patterns according to their $SupMax2$ value and then chose the top 70 patterns and created the heatmap for them. Figure 4.2 a, b, and c present the results for the three phenotypes with a value of $\delta = 0.5$

The black cells in the figures indicate a value of 1 and the white cells indicates a value of zero. Each row represent a pattern and each column represents a graph. It is clear from the heatmap that the columns of the graphs annotated with the phenotype are mostly black cells and the columns of graphs annotated with the background class are mostly white cells. The heatmap shows the ability of our algorithm in detecting and reporting patterns that can be used for graph classification with a very high accuracy.


a) Phenotype "Leukocytes"


b) Phenotype "Leukocytes, Mononuclear"



c) Phenotype "Neoplasms"

Figure 4.2. Heatmap for top 70 patterns (δ= 0.5)

4.3.4. Enrichment Anaylsis

Biological enrichment analysis was performed on the patterns reported by our algorithm to gain more information about the biological significance of the reported patterns. For the enrichment analysis, we used **D**atabase for **A**nnotation, **V**isualization, and **I**ntegrated **D**iscovery - DAVID [8, 9]. We also used **clusterProfiler** a library in R programming language that is used for analyzing and visualizing functional profiles (GO and KEGG) of gene and gene clusters [24]. We used DAVID for GO term and KEGG enrichment analysis and used **clusterProfiler** library for disease enrichment analysis in the reported patterns.

Figure 4.3 presents our results for the percentage of enriched patterns in GO biological processes terms for the two values of δ. From Figure 4.3 we can see that percentage of the enriched patterns is very high. The enrichment percentage is more than 90% for the phenotype "Leukocytes" and "Leukocytes, Mononuclear" at $\delta = 0.4$ and even for the third phenotype "Neoplasms" it is close to 60% which is still considered a high percentage. As the value of δ increases, the number of patterns decrease which maybe the cause for the enrichment percentage drop as it appears in the figure. We noticed that there is a significant drop in the enrichment percentage for the phenotype "Neoplasms". When we investigated the reason for this drop, we found that the number of patterns reported for that phenotype at $\delta = 0.5$ is only 25 patterns with an average size of 3 nodes and 3 edges which means the patterns are relatively small patterns. The low number and small size of the patterns can explain the drop in the enrichment percentage.

Figure 4.4 presents the results of the KEGG enrichment analysis. By looking at the results, we can see the percentage of the enriched patterns at a value of $\delta = 0.4$ about 90% for phenotype "Leukocytes" and very close to 100% for the phenotype "Leukocytes, Mononuclear". When we increase δ to 0.5 the enrichment drop to about 70% for both phenotypes which is still considerably high. As for the phenotype "Neoplasms" is it very low at both values of δ which can because of the same reasons mentioned above that caused the drop in the GO terms enrichment.

We also analyzed the top GO terms in the reported patterns for every phenotypes and created a heatmap for the top 10 GO terms enriched in the patterns. Figure 4.5, 4.6, and 4.7, presents the top 10 GO terms significantly enriched in the patterns reported by our algorithm. The rows represent the GO terms and the columns are the patterns. A black cell indicates that

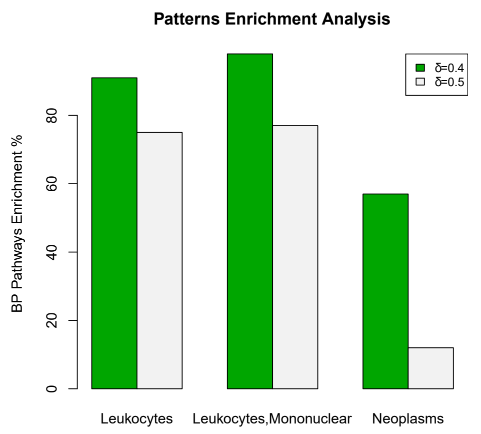


Figure 4.3. Patterns Biological Processes Enrichment Analysis.

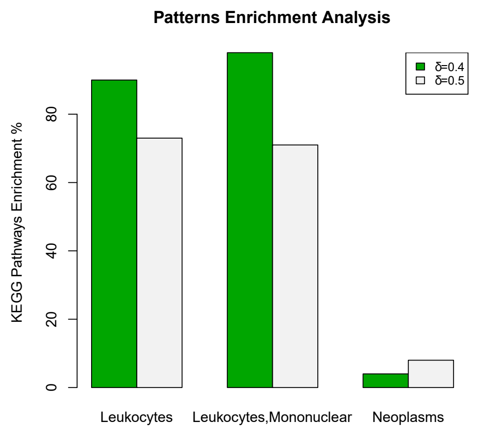


Figure 4.4. Patterns KEGG Pathways Enrichment Analysis.

the GO term is enriched in the corresponding pattern. the GO biological processes terms include *ribonucleoprotein complex biogenesis*, *ribosomal small subunit biogenesis*, *rRNA metabolic process*, *antigen receptor-mediated signaling pathway*, *positive regulation of lymphocyte activation*, *immune response-activating cell surface receptor signaling pathway*, and *regulation of cell activation.*

Moreover, we checked the disease enrichment for the patterns and reported the top 5 diseases enriched in the reported patterns for every phenotype. The results is summarized in Table 4.3. Other disease enriched in the patterns include *papillary thyroid carcinoma*, *thyroid carcinoma*, *hypersensitivity reaction type IV disease*, *autoimmune disease of endocrine system*, *corneal disease*, and *dermatitis.*

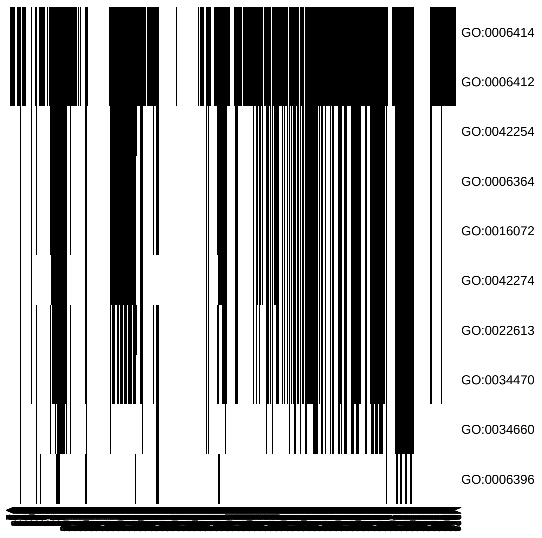


Figure 4.5. Heatmap for top 10 GO terms enriched in the phenotype (Leukocytes).

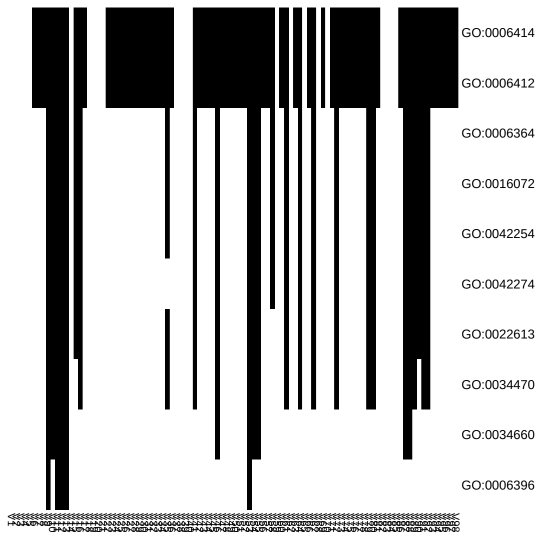


Figure 4.6. Heatmap for top 10 GO terms enriched in the phenotype (Leukocytes, Mononuclear).

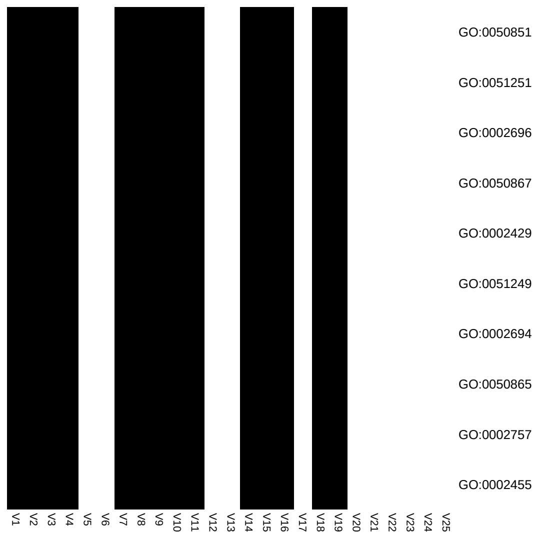


Figure 4.7. Heatmap for top 10 GO terms enriched in the phenotype (Neoplasms).

Table 4.3. Top 5 diseases enriched in the second algorithm's phenotypes patterns.

Phenotype	Top Diseases
Leukocytes	• ischemia • malignant glioma • keratosis • integumentary system disease • congenital hypoplastic anemia
Leukocytes, Mononuclear	• malignant glioma • ischemia • agranulocytosis • autosomal recessive disease • congenital hypoplastic anemia
Neoplasms	• chronic leukemia • lupus erythematosus • sarcoidosis • acute lymphocytic leukemia • aplastic anemia

5. CONCLUSION AND FUTURE WORK

Genes coexpression networks classification is a very important problem with many applications like unknown genes functional annotation and discovery of new genes functions. In this thesis, we propose algorithms for gene coexpression networks classification using discriminative patterns mining.

The first algorithm propose the use OR function for measuring the discriminative power of pattern other than the traditional AND function. It uses a greedy search algorithm to minimize the search space of a very large graph dataset with millions of edges. The second algorithm uses an antimonotone discriminative function for mining patterns. It utilize a number of pruning strategies to prune the search and remove any redundant patterns.

We compared the two algorithms with two baseline algorithm for graph classification. The first baseline algorithm uses a simulated annealing function for discriminative pattern mining and the second baseline algorithm uses the graph's topological attributes as features for classification. Our two algorithms outperformed both of the baseline algorithms and shows a very accurate classification results.

Future work can address improving the discriminative functions used to evaluate the patterns. In the first algorithm, we used AND and OR functions to measure the discriminative power of a pattern. A future work may evaluate how the use of other different functions like XOR for example can affect the results. For example, sometimes for a cellular function it requires the presences of specific genes and also the absence of other genes, using XOR function might be suitable to represents these situations. This could lead to a discovery of new and complex relationships between genes and provide a very helpful insights on how the genes work together in the cell.

REFERENCES

[1] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. Classification and regression trees. *Wadsforth International Group*, 1984.

[2] P. Cunningham and S.J. Delany. k-nearest neighbour classifiers. technical report. *Artificial Intelligence Group*, 2007.

[3] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[4] G. Fang, G. Pandey, W. Wang, M. Gupta, M. Steinbach, and V. Kumar. Mining low support discriminative patterns from dense and high-dimensional data. *IEEE TKDE*, 2011.

[5] J. Friedman. On bias, variance, 0/1 loss, and the curse of dimensionality. *Data Mining and Knowledge Discovery*, page 55–77, 1997.

[6] B. Goethals. Survey on frequent pattern mining. *Manuscript*, 2003.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[8] D.W. Huang, B.T. Sherman, and R.A. Lempicki. Systematic and integrative analysis of large gene lists using david bioinformatics resources. *Nature protocols*, 4(1):44–57, 2008.

[9] D.W. Huang, B.T. Sherman, and R.A. Lempicki. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic acids research*, 37(1): 1–13, 2009.

[10] E. Hunt, J. Martin, and P. Stone. Experiments in induction. *New York: Academic Press*, 1966.

[11] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105, 2013.

[12] M. Koyutürk, A. Grama, and W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *ISMB/ECCB'04*, pages i200–i207, 2004.

[13] G. Li, M. Semerci, B. Yener, and M.J. Zaki. Graph classification via topological and label attributes. In *Proceedings of the 9th International Workshop on Mining and Learning with Graphs (MLG), San Diego, USA*, 2011.

[14] O. Mason and M. Verwoerd. Graph theory and networks in biology. *IET Systems Biology*, 1 (2):89–119, March 2007.

[15] MR. Mehan, J. Nunez-Iglesias, M. Kalakrishnan, MS. Waterman, and XJ. Zhou. An integrative network approach to map the transcriptome to the phenome. *J Comput Biol*, page 16(8):1023–1034, 2009.

[16] MR. Mehan, J. Nunez-Iglesias, C. Dai, MS. Waterman, and XJ. Zhou. An integrative modular approach to systematically predict gene-phenotype associations. *BMC Bioinformatics*, page 11(Suppl 1):S62, 2010.

[17] T.N. Phyu. Survey of classification techniques in data mining. *International Multiconference of Engineers and Computer Scientists,Hong Kong*, 2009.

[18] L. Ralaivola, S J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18:1093–1110, 2005.

[19] M. Soundarya and R. Balakrishnan. Survey on classification techniques in data mining. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(7): 7550–7552, 2014.

[20] J.M. Stuart, E. Segal, D. Koller, and S.K. Kim. A genecoexpression network for global discovery of conserved genetic modules. *Science*, 302(5643):249–255, 2003.

[21] M. Thoma, H. Cheng, A. Gretton, J. Han, H.P. Kriegel, A.J. Smola, L. Song, P.S. Yu, X. Yan, and K.M. Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *9th SIAM Conf. on Data Mining (SDM 2009), Sparks, NV*, pages 1075–1087, 2009.

[22] V.N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, 1995.

[23] X. Yan, H. Cheng, J. Han, and P.S. Yu. Mining significant graph patterns by scalable leap search. *SIGMOD*, 2008.

[24] G. Yu, L. Wang, Y. Han, and Q. He. clusterprofiler: an r package for comparing biological themes among gene clusters. *OMICS: A Journal of Integrative Biology*, 16(5):284–287, 2012.

[25] B. Zhang and S. Horvath. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(1):17, 2005.