ALGORITHMS FOR MINING MAXIMAL QUASI FREQUENT SUBNETWORKS

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

By

Eihab Salah El Radie

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

Major Department:
Computer Science

April 2018

Fargo, North Dakota

NORTH DAKOTA STATE UNIVERSITY

Graduate School

Title

ALGORITHMS FOR MINING MAXIMAL QUASI FREQUENT SUBNETWORKS

By

Eihab Salah El Radie

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

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Saeed Salem
Chair

Dr. Simone Ludwig

Dr. Mukhlesur Rahman

Approved:

09 November 2017
Date

Prof. Kendall E. Nygard
Department Chair

ABSTRACT

Frequent graph mining has received considerable attention from researchers. Existing algorithms for frequent subgraph mining do not scale for large networks, and take hours to finish. Mining multiple gene coexpressions networks allows for identifying context-specific modules. Frequent subnetworks represent essential biological modules.

In this thesis, we propose two algorithms for mining frequent subgraphs. In the first algorithm, we propose a parallel algorithm for mining maximal frequent subgraphs from gene coexpression networks. Despite the algorithm's parallelization, it takes much time and it does not allow relaxation. This inspired us to develop a second algorithm that solves those problems. In the second algorithm, we propose a greedy approach for mining approximate frequent subgraphs. Experiments on real tissue-specific RNA-seq expression networks and synthetic data demonstrate the effectiveness of the proposed algorithms. Moreover, biological enrichment analysis shows that the reported patterns are biologically relevant and enriched with known biological processes and KEGG pathways.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my advisor, Dr. Saeed Salem. Only by his leadership I was able to produce this work. Without his continuous optimism concerning this work, enthusiasm, encouragement and support, this study would hardly have been completed. His wisdom, knowledge and commitment to the highest standards inspired and motivated me. I would also like to thank my committee members, Dr. Simone Ludwig and Dr. Mukhlesur Rahman, for taking the time to serve on my thesis committee and helping me to graduate from the University.

I also want to express my thanks for North Dakota State University and the Department of Computer Science for offering me the opportunity to study at such a great school. The people and the environment have made it a great place to study and learn.

Finally, I would like to thank my family, specially my mother and my wife, who encouraged me and prayed for me throughout the time of research, the time of writing this thesis, and my life in general.

DEDICATION

This thesis is dedicated to the memory of my father, Salah El Radie. I miss him every day, but I am glad to know he saw this process through to its completion, from heaven. Making him proud of me gave me all the strength and the willingness to success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Advances in data collection and generation have led to enormous datasets being collected in various scientific domains, such as biological networks, transportation networks, and social networks.. Due to rapid data growth and integration in biological areas, new technologies are making it easier and cheaper to perform experiments that generate tremendous quantities of data for biological and medical related studies [9]. Moreover, Social Networks became so popular which serve as rich sources of data where billions of instant messages are exchanged every day, millions of users exist, and billions links exist among them [3]. The abundance of social networks raises the need for new techniques to store and analyze this data. This has attracted the researchers' attention to find ways to extract the relationships among entities and to use data mining techniques and methods to discover useful and meaningful information out of this data.

Graphs have emerged as natural data structures to model the interactions in complex systems, e.g., social networks, chemical compounds, protein 3D structure, biological pathways, coexpression networks, and protein-protein interaction networks. An example illustrating how graphs model interactions in biological systems is protein-protein interaction (PPI) network. In PPIs, a protein is represented by a vertex (node) and the interaction between various proteins are represented by the links (edges) between the vertices. Another example is social networks, like Twitter, which can be easily modeled by a graph; a vertex represent a profile and an edge between vertices represent a friendship connection or a relation. Analysis of the data, after it is modeled into graphs using data mining techniques, makes a positive impact on marketing when working on social networks.

Frequent pattern mining (FPM) has been studied extensively to solve hard problems. FPM is a fundamental data mining technique along with association rules, clustering, classification, etc. Several approaches have been proposed for the discovery of frequent subgraphs and motifs in networks. Researchers work on finding the patterns that appear frequently in these datasets. Such patterns include frequent itemsets, frequent sequences, frequent dense subgraphs, and maximal frequent subgraph patterns. Thus far, most of the existing methods have focused on the analysis of individual biological networks. However, a single network is insufficient to find patterns with

multiple signals. This urged the scientists to develop methods that support the integrative analysis of multiple biological networks [33].

Clustering is the most commonly used technique of data mining [26]. The task of graphs clustering is done based on their structural similarity. Since several techniques have been proposed to discover frequent patterns in large datasets, researchers took advantage of those techniques for clustering graphs based on the occurrence of patterns in these graphs.

In recent years, graph classification have received a lot of researchers' attraction due to its application in wide range fields. For example, in chemistry, chemical compounds can be represented as graphs and graph classification techniques can be applied to predict the toxicity of these compounds [38]. Coexpression networks classification involves a dataset of multiple graphs labeled with a class label, e.g., tissue-specific gene coexpression networks, and to predict the tissue that the network belongs to.

Graph mining techniques are often used in systems like Amazon and Netflix which take advantage of the interactions in social networks to improve personalized advertising as well as making suggestions based on the tastes and preferences of your friends. In bioinformatics, network analysis has made a huge impact on health care. In System Biology, research has shown that discovering the unknown complexes in proteins networks by observing the interactions is important.

1.1. Contribution

Our first contribution in this thesis is a multi-threaded implementation of an efficient algorithm for detecting frequent subgraphs in biological networks [28]. On large graph such as gene coexpression networks that have millions of coexpression links, the Mule algorithm [28] takes hours to days depending on the minimum support threshold. With the availability of multi-core processors for personal machines, it is important to design parallel implementation of frequent subgraph mining algorithms. Our proposed parallel implementation has been published:
E. E. Radie and S. Salem, "A parallel algorithm for mining maximal frequent subgraphs," 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Kansas City, MO, USA, 2017, pp. 1965-1971.

Our second contribution in this thesis is an algorithm for mining quasi frequent coexpression subnetworks. The first algorithm takes much time even after it is being parallelized and it also allows no relaxation while mining the subgraphs. These reasons have encouraged us to think

of an algorithm to overcome the first algorithm's challenges. Our proposed algorithm has been published:

E. E. Radie and S. Salem, "Mining quasi frequent coexpression subnetworks," 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Kansas City, MO, USA, 2017, pp. 1736-1740.

1.2. Thesis Overview

For the remainder of this thesis, we begin by presenting important topics that are related to our work in Section 2. In Section 3, we introduce our first proposed algorithm including the problem description, the methodology, and the experiments. In Section 4, we introduce our second proposed algorithm beginning by problem description along with defining important terms, then we move to the details of the proposed algorithm, followed by showing the performed experiments and results. Finally, in Section 5, outlines the conclusion and future work for further improvements on our proposed methods.

2. RELATED WORK

Mining frequent patterns is the task of discovering the repetitive substructures in the database. Usually, scholars are interested in frequent substructures that appear in transactions of the database. This technique is widely used as it is one of the basic problems in data mining. Generally, frequent pattern mining comes in different flavors, including: itemset mining, sequence mining, graph pattern mining, and pattern and rule assessment.

In this section, we present the related work with a brief background about important topics in two categories of frequent pattern mining approaches: 1. Itemset Mining - discovering frequent subsets in a set of items. 2. Graph Mining - finding frequent subgraphs in a dataset of graphs. Moreover, we present a brief background about topics that overlap with frequent pattern mining such as: clustering, and classification techniques.

2.1. Itemset Mining

The first problem that inspired scientists to start the field of frequent pattern mining was the need to mine frequent itemsets and associations between the items. When talking about this problem, usually the prototypical application is market basket analysis, i.e., to analyze the customers shopping carts, also called market baskets, to find which sets of items are frequently bought together at the market.

Let $\mathcal{I}$ be the set of all elements called *items*, $\mathcal{I} = \{x_1, x_2, ..., x_m\}$. An itemset X of size k is called *k-itemset*, where $X \subseteq \mathcal{I}$. Let $\mathcal{T} = \{t_1, t_2, ..., t_n\}$ be another set of elements called transaction identifiers or *tids*. A *transaction* is a tuple $T = (t, X)$, where $t \in \mathcal{T}$. A transactions database is a set of transactions [13]. The *support* of an itemset X in a dataset D, denoted $sup(X, D)$, is the number of transactions in D that contain the itemset X. An itemset X is called *frequent* in D if $sup(X, D) \geq minsup$, where $minsup$ is a user-specified minimum support threshold. Figure 2.1 shows an example of a transactions database with five transactions and five unique items. In the example, after finding all the itemsets, there were no itemsets that appear in whole number of transactions. We used the minimum support threshold, 3, and found eight frequent itemsets.

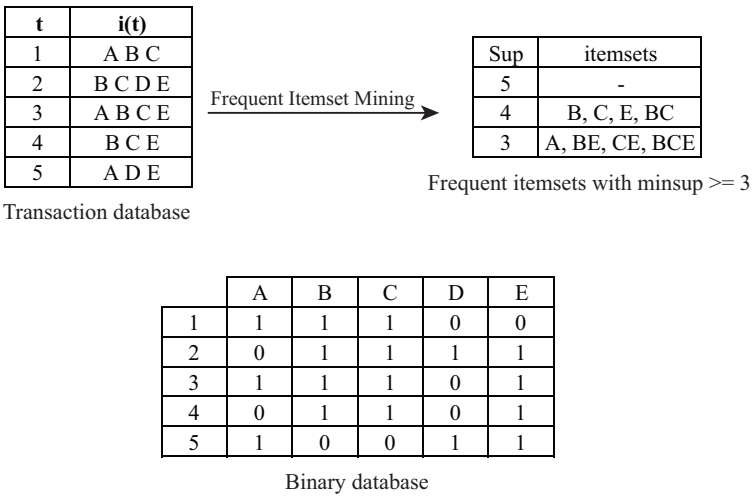


Figure 2.1. Example of a frequent itemset mining

2.1.1. General Itemset Mining Algorithms

One of the earliest efficient techniques to find association rules between large itemsets is the Apriori algorithm proposed by Agrawal and Srikant [1]. The algorithm presents a technique to discover all association rules from transaction data and to find all sets of items (itemsets) that have transaction support above minimum support. The Apriori algorithm is classified as breadth-first enumeration method as it employs a level-wise exploration of the itemset search space. As no superset of an infrequent itemset can be frequent, it prunes all supersets of any infrequent candidate.

The general idea of Apriori is to generate candidate patterns of length $k+1$ by finding the intersection between two patterns of length k, using the breadth-first enumeration tree structure. The two combined patterns share $k-1$ subpatterns.

After applying Apriori algorithm on the example in Figure 2.1 using $minsup = 3$, Figure 2.2 shows the itemset search space for the Apriori algorithm. Each node contains an itemset along with its support, e.g., $AB(2)$ indicates that support of AB is 2. Two itemsets are connected if one is a prefix and immediate subset of the other. Apriori enumerates the candidate patterns in a level-wise manner, which proves the power of pruning in the search space. For example, once we determine that item D is infrequent, we can prune any itemset that has D as a prefix, i.e., the entire subtree under D can be pruned. Also, the extension $\{BCD\}$ from $\{BC\}$ can be pruned, since $\{CD\}$ is infrequent.

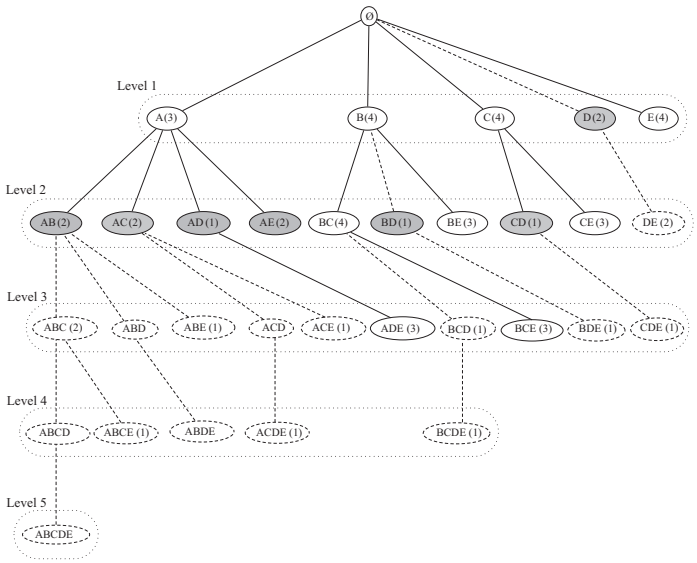
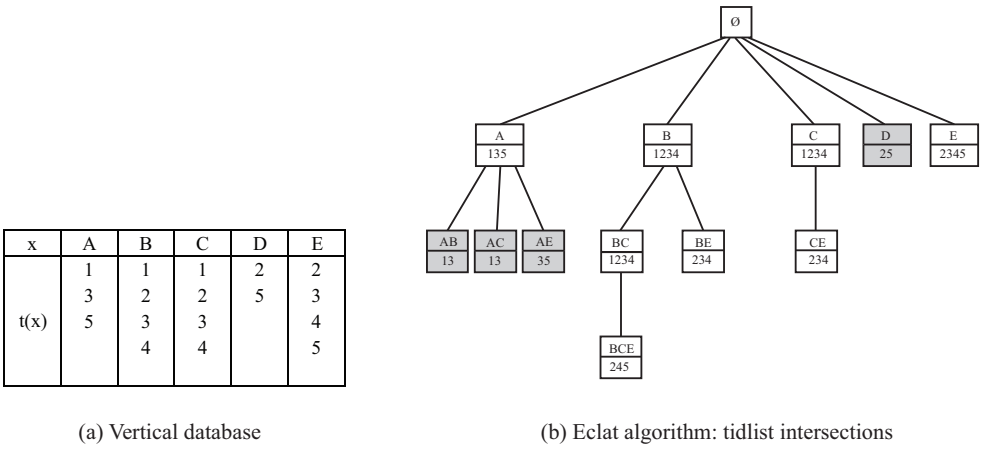


Figure 2.2. Apriori Enumeration tree of Figure 2.1(a). Shaded nodes indicate infrequent itemsets, Solid lines indicate frequent, whereas dashed nodes and lines indicate all of the pruned nodes and branches.

After the Apriori algorithm, many researchers proposed approaches to improve the efficiency and performance. Zaki et al. [46] proposed the Eclat algorithm. The algorithm is a depth-first-based algorithm that leverages the tidsets directly for support the computation. It employs a vertical representation of the binary database D and it requires to go over the database only once for enumerating all the association rules. The basic idea of the Eclat algorithm is that the support of a candidate itemset can be computed by intersecting the tidsets of suitably chosen subsets [45]. Figure 2.3(a) shows the vertical representation of the example in Figure 2.1. Also it illustrates the Eclat algorithm application on the same example. With $minsup = 3$, the initial prefix equivalence class will not contain the subset D since $sup(D) = 2$. The next step is that Eclat intersects $t(A)$ with each of $t(B)$, $t(C)$, and $t(E)$ to obtain the tidsets for $\{AB, AC, AD$ and $AE\}$ but all these will be pruned as they are infrequent (marked gray). For node B, Eclat intersects $t(B)$ with each of $t(C)$, $t(E)$ and it keeps them both as they both are frequent, and so on.

Frequent itemsets mining result in an enormous number of frequent itemsets that are highly overlapping. The use of condensed representations will reduce the size of the output without any information loss and the computational demands.

Maximal frequent itemset let $\mathcal{F}$ denote the set of all frequent itemsets. A frequent itemset $X \in \mathcal{F}$ is called *maximal* if it has no frequent supersets. Let $\mathcal{M}$ be the set of all maximal



(a) Vertical database

(b) Eclat algorithm: tidlist intersections

Figure 2.3. Eclat Enumeration tree of Figure 2.1(a)

frequent itemsets, the set $\mathcal{M}$ is a condensed representation of the set of all frequent itemset $\mathcal{F}$. Also, it can make it easier to analyze the mined patterns. Gouda and Zaki proposed the GenMax algorithm [15], a backtrack search based algorithm for mining maximal frequent itemsets. The set of maximal frequent itemsets of the example in Figure 2.1 is $\mathcal{M} = \{A, BCE\}$.

2.2. Graph Mining

In the recent years, many algorithms have been proposed to discover the frequent subgraphs and motifs in networks. This is due to the generation of vast amount of data every single day and it became available to be processed to find the relationships between those subgraphs as well as the differences. Social networks, biomolecular interaction networks, and transportation networks are examples of graph datasets. Researchers developed efficient algorithms for counting the frequent small subgraphs in a graph which has millions of nodes and thousands of millions of edges. To develop an algorithm that finds all subgraphs in a graph dataset is computationally expensive.

In this section, we present several graph mining problems related to our work. The algorithms made for these problems are different based on the type of input data.

2.2.1. Frequent Subgraph Mining

Frequent subgraph mining is the process of finding frequent subgraphs given a graph dataset. A dataset can be one large single graph or a set of multiple graphs. Since different data types like text, images, videos, or data like genes coexpressions or chemical compounds can be easily represented as graphs, frequent subgraph mining became a very popular research problem [22].

Like the minimum support threshold in itemset mining, a subgraph is called *frequent* if it appears in a number of graphs that is greater than or equal to a user defined support threshold.

Cook and Holder were one of the first researchers who proposed algorithms in detecting substructures in graph data as a part of their system SUBDUE [10]. SUBDUE finds substructures that compress the original data and represents structural concepts in the data. It also can produce a hierarchical description of the structural regularities in the data.

Inokuchi et al. [21] proposed the AGM algorithm for mining all frequent induced subgraphs by following a breadth-first approach that grows the subgraphs by adding one vertex at a time. The algorithm proposed an efficient method to mine the association rules among the frequent subgraphs in a graph dataset. For the algorithm to work, a graph transaction is represented by an adjacency matrix. Also, they used an extended algorithm of the basket analysis to mine the frequent patterns that is appearing in the matrices.

Kuramochi and Karypis [31] proposed the FSG algorithm for mining all frequent connected subgraphs by following a pattern growth level-by-level approach that extends subgraphs by adding an edge. The FSG algorithm proposed optimizations for candidate generation and frequency counting and introduced effecient canonical labeling.

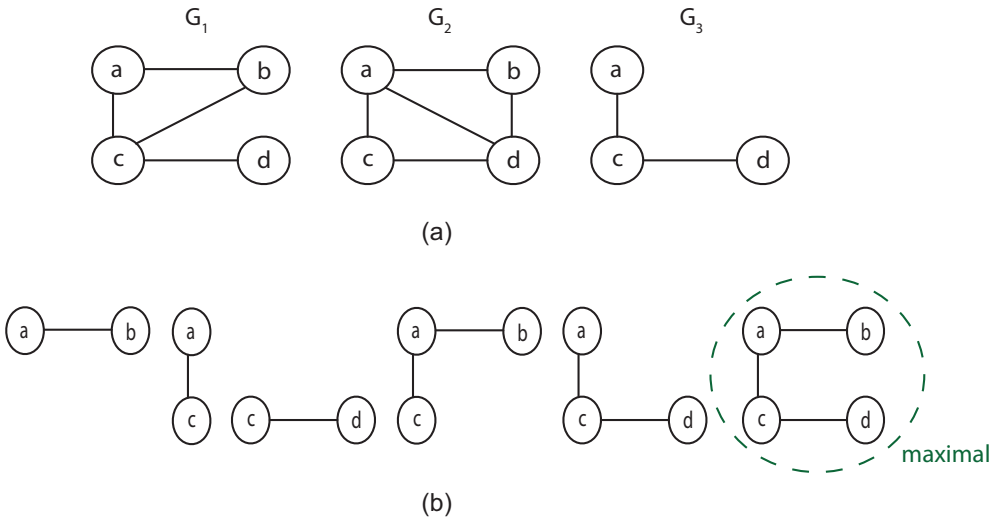


Figure 2.4. Example of frequent subgraph mining using minimum support = 2

AGM and FSG, Apriori-based algorithms, have two challenges: (i) candidate generation: to generate $(n+1)$ substructure candidates from frequent n-subgraphs is computational costly; and

(ii) to prune false positive. Yan and Han [44] proposed the gSpan algorithm that follows a depth-first pattern growth approach, to overcome the obstacles in AGM and FSG, and they succeeded to reduce the significant cost of challenges in frequent subgraph mining that used to exist in both algorithms.

Nijssen and Kok [34] proposed the Gaston tool that mines frequent subgraphs starting from patterns with simpler complexity, i.e., sequences, and trees. These algorithms are computationally extensive due to the graph and subgraph isomorphism checking [12], and take hours and even days on large datasets with low minimum support threshold. These algorithms were proposed for general labeled graphs that allow the same node/edge label to appear for multiple vertices/edges in the same graph.

In bioinformatics, some networks such as protein-protein interaction networks and gene coexpression networks have unique node labels. More efficient algorithms have been proposed for the analysis of graphs with unique labels. The Mule algorithm [28, 29] proposed novel enumeration approaches for all maximal and closed frequent subgraphs, respectively. The Mule algorithm proposed an efficient enumeration tree in which each frequent subgraph appears exactly once, thus eliminating the need for duplicate checking. The algorithm is a depth-first search that allows for deciding whether a subgraph is maximally frequent.

Algorithms for mining closed and maximal frequent subgraphs are very important compared to algorithms that find all the frequent subgraphs in a dataset. This is due to the reason that the reduction in the size of the output will not cause any information loss, which also makes it easier to analyze the mined patterns. Moreover, it reduces the computational demands.

Table 2.1. Comparison between frequent, close, and maximal patterns

minsup	*Frequent*	*Closed*	*Maximal*
18	1,776,157	14,083	9,038
19	599,798	9,862	6,463
20	214,119	7,055	4,701
21	93,736	5,157	3,570
22	44,134	3,829	2,711

Table 2.1 shows the huge difference in number of frequent patterns compared to the number of closed patterns and number of maximal. We performed the experiment on a dataset [36] that has 35 coexpression networks inferred from 35 tissues. For example, in the table, for using minimum support = 18 there will be more than a million and a half reported frequent patterns, while this number massively decreases to nearly 14 thousand closed frequent patterns and 9 thousand maximal frequent patterns.

Theoretically, frequent pattern mining in graph data are more difficult than frequent pattern mining in any other structured data, i.e., itemset mining, sequence mining, as well as tree mining.

2.2.2. Quasi Frequent Subgraph Mining

The definition of *quasi frequent subgraph* does not mean each edge of the subgraph should appear in a restricted number of graphs as in regular frequent subgraphs. It requires each edge of subgraph to appear in at least t of the supporting graphs, i.e., an edge can still appear in a smaller number of graphs and yet be part of a quasi frequent subgraph. Figure 2.5(b) shows an example of quasi frequent subgraph where the subgraph $\{a-b, a-d, a-e, b-c\}$ has been extended with two edges that are frequent in 2 out of the 3 graphs in which the seed subgraph is frequent. The edge occurrence matrix in Figure 2.5(c) shows how the edges are added respectively to form the subgraph.

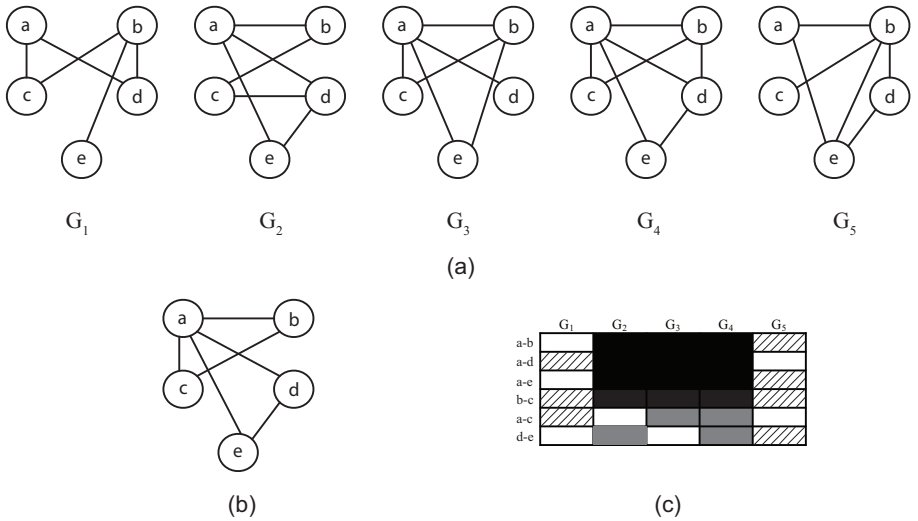


Figure 2.5. Examples of the two-step approach for mining a representative set of quasi frequent subgraphs (a) Graph dataset with five graphs. (b) Extension of seed {a-b, a-d, a-e, b-c} with two edges. (c) Edge occurrence matrix of subgraph in (b)

Researchers proposed methods that have mined frequent pattern from the edge occurrence in the graphs and then partitioned these pattern into connected subnetworks. Huang et al. [18] employed simulated annealing to mine semi-frequent edgesets that frequently appear in the coexpression networks. The result was edgesets that are not entirely connected, then they extracted all the quasi frequent subgraphs that are highly connected.

2.2.3. Clustering Approaches for Mining Graphs

Researchers studied the building principles of biological networks in attempt to potentially revolutionize the view of biology and disease pathologies [4]. The popular clustering approach usually extracts densely connected modules from biological networks, which are often biologically related and meaningful, e.g. a dense PPI subnetwork may correspond to a protein complex [43].

Researchers like in [32, 16] used to build summary graph from the coexpression graphs by the coexpression links that appear in at least a minimum number of coexpression networks, where the support threshold is user-specified. Figure 2.6 shows the summarization process. Figure 2.6(a) shows a dataset of multiple coexpression graphs, while in (b) the summary matrix of the dataset. In Figure 2.6(c), the summary graph contains the edges that appears in at least 3 graphs.

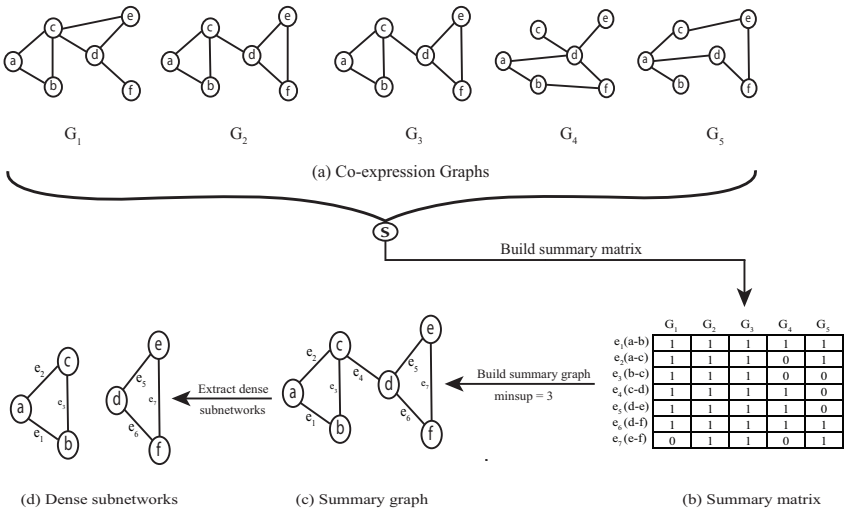


Figure 2.6. Extracting dense subnetworks from summary graph

2.3. Clustering and Subspace Clustering

Clustering is the approach of finding groups of similar objects within dataset while keeping the noise away or in different groups. Algorithms for these approaches always had the challenge

that the data being high-dimensional because many of the dimensions are often irrelevant. The irrelevant dimensions might hide clusters in noisy data which confuse the clustering algorithms.

The challenge of dealing with multi-dimensional data is known as *the curse of dimensionality*. Unlike the the traditional approaches, subspace clustering algorithms focus the search process for the relevant dimensions which make it tolerant to find clusters that exist in multiple or overlapping subspaces. Figure 2.7 present an illustration of subspaces overlapping. For example, a group of people who have $70 \leq Salary \leq 95$ and $22 \leq Age \leq 40$, and another group who have $40 \leq Salary \leq 85$ and $35 \leq Age \leq 50$ are overlapped, i.e., they both contain the same people who get salary in the range 70 and 75 and their age between 35 and 40.

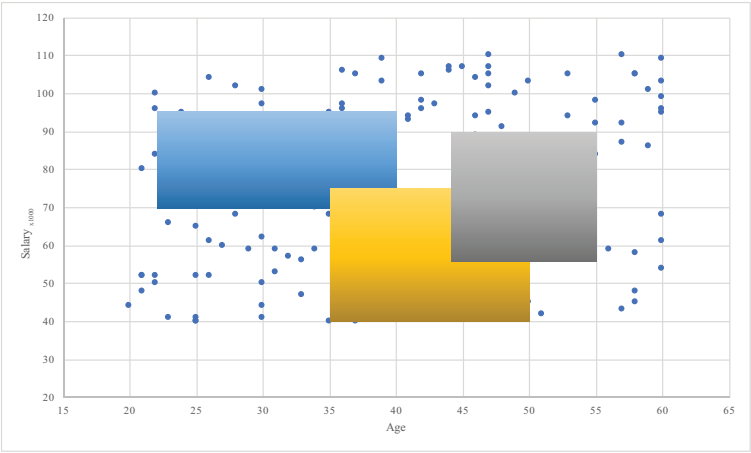


Figure 2.7. Example of subspace clusters.

Subspace clustering is often described as an extension of the traditional clustering, which seeks to find clusters in different subspaces within a dataset [35].

Subspace clustering comes in two flavors based on the search strategy. **Top-down** algorithms that find initial clustering in high dimensional data and then improve the result by evaluating the subspaces of each other iteratively. **Bottom-up** algorithms that form clusters by combining dense areas in low dimensional spaces.

Zait et al. [47] offered a comprehensive study of clustering algorithms. They proposed a methodology for comparing clustering methods based on two metrics. The most significant metrics to be used are: the quality of the clustering result, and the performance of the execution of each algorithm. Morever, subspace clustering has been reviewed in recent book by Berhkin

[5]. This survey focuses on clustering algorithms from data mining perspective. Kriegel et al. [30] discussed the problems motivating subspace clustering and presented a handful number of exemplary algorithmic solutions. They also sketched different definitions and usages of subspaces for clustering.

2.4. Classification

Classification refers to the technique of predicting a class label for a given unlabeled data instance. Generally, there is a labeled dataset and the objective is to predict the label of each unlabeled instance by using information from the labeled dataset. Figure 2.8 shows an illustraion of the classification task in general. Many approaches and techniques have been proposed classify data, including K-Nearest Neighbor, Decision Trees, Naive Bayes, and Support Vector Machines (SVM).

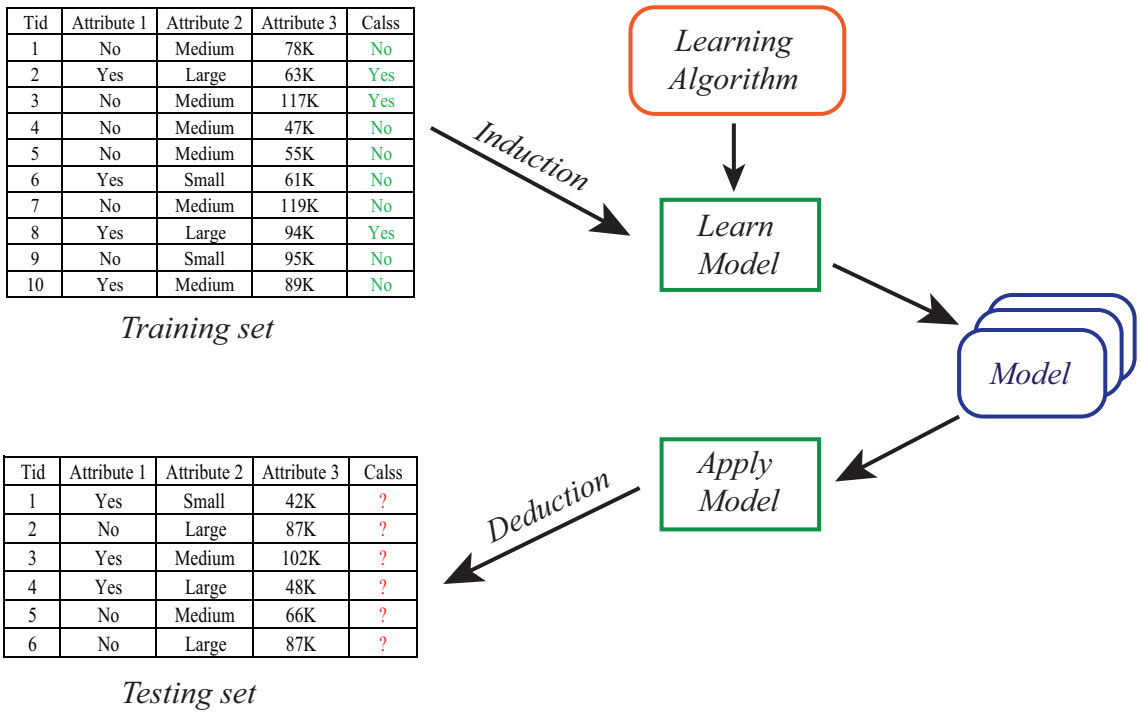


Figure 2.8. Classification task illustration

2.4.1. k-Nearest Neighbor

Perhaps the most straightforward classifier in machine learning techniques and algorithms is the Nearest Neighbour Classifier [11]. In a K-Nearest Neighbor classifier, the classification is achieved by identifying the class labels of its closest k neighbors, where k is the number of neighbors, then it will predict class label of the data instance as the label of the majority of the neighbors (majority vote).

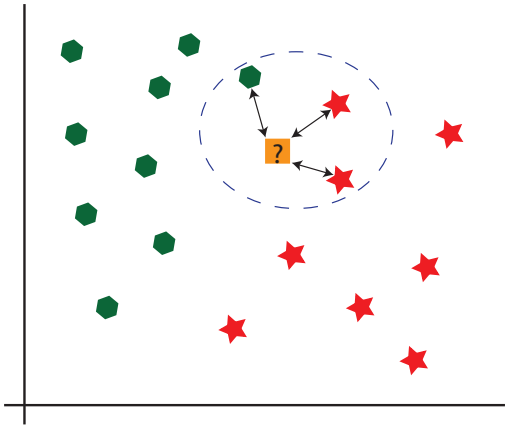


Figure 2.9. Example of K-NN classification with $k = 3$

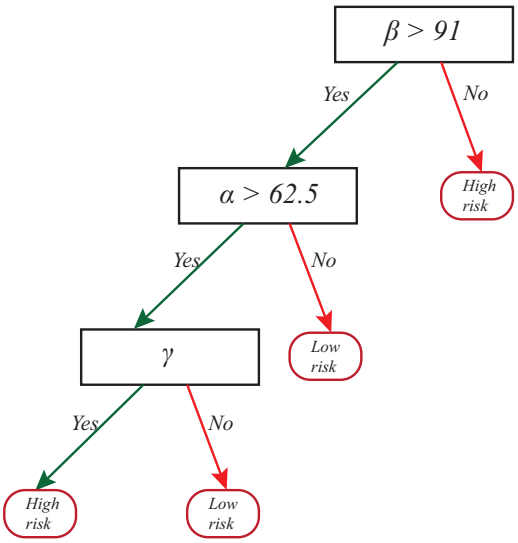
Two metrics should be determined before running the algorithm: the value of k, and the type of distance used. The large value of k, the less classification accuracy due to taking into consideration far neighbors which affect the majority vote value. On the other hand, choosing small value of k can also reduce the accuracy when there are some noisy data instances close to the targeted data instance. For the second metric, depending on the type of the dataset, we can choose which distance method to use, e.g., the Euclidean distance, the Manhattan distance, or any other method from Minkowski distance methods.

Figure 2.9 shows an example of K-NN classifier that uses $k = 3$. The closest three shapes are two stars and a polygon. Since the majority of the three neighbors are stars, then the shape will be labeled as a star.

One of the challenges of K-NN classifier is the calculation of distances for point, i.e., the K-NN classifier does not build a model one time and use that model for every data instance. It builds a model for every instance separately, that why it is called *LazyLearning* classifier.

2.4.2. Decision Trees

Decision trees classify instances by sorting them based on feature values. Decision Trees classifier uses a tree structure learning to classify data instances. The leaves in a decision tree represent class labels and branches represent conjunctions of features that lead to the class labels [42]. The root is chosen so its feature split the tree the best. Finding a way to choose the root that best divide the data instances is proposed in many researchers' work, such as gini index [6], and information gain [19].



α	β	γ	Risk?
54	98	Yes	Low
85	90	No	High
72	105	Yes	High

a) Training model

b) Test data

Figure 2.10. Example of decision tree classification

A big advantage for decision trees is that the classifier generated is highly interpretable. Figure 2.10 presents medical example where patients are classified into one of two classes: high risk or low risk. High risk patients would not survive at least 30 days based on the initial 24-hour data. Figure 2.10(a) shows a training model based on measurements taken from 19 patients during the first 24 hours. The attributes are: 1. Age (α). 2. The minimum systolic blood pressure over the initial 24 hours period (β). 3. Sinus tachycardia presence (γ). The model is applied on the test data shown in Figure 2.10(b) and predicted the class label with high precision.

3. A PARALLEL ALGORITHM FOR MINING MAXIMAL FREQUENT SUBGRAPHS[1]

3.1. Introduction

Several approaches have been proposed for the discovery of frequent subgraphs and motifs in network. This interest has been motivated by the vast amount of graph data that has been collected in a diverse set of domains, including social networks, biomolecular interaction networks, and transportation networks.

Frequent pattern mining has been proposed by Agrawal et al. [1] who introduced an Apriori-based algorithm to discover all association rules. Several algorithms for frequent graph mining have adopted the Apriori principle. Inokuchi et al. [21] proposed the AGM algorithm for mining all frequent induced subgraphs by following a breadth-first approach that grows the subgraphs by adding one vertex at a time. Kuramochi and Karypis [31] proposed the FSG algorithm for mining all frequent connected subgraphs by following a pattern growth level-by-level approach that extends subgraphs by adding an edge. The FSG algorithm proposed optimizations for candidate generation and frequency counting and introduced effecient canonical labeling. Yan and Han [44] proposed the gSpan algorithm that follows a depth-first pattern growth approach. These algorithms were proposed for general labeled graphs that allow the same node/edge label to appear for multiple vertices/edges in the same graph. Nijssen and Kok [34] proposed the Gaston tool that mines frequent subgraphs starting from patterns with simpler complexity, i.e., sequences, and trees. These algorithms are computationally extensive due to the graph and subgraph isomorphism checking, and take hours and even days on large datasets with a low minimum support threshold.

In bioinformatics, some networks such as protein-protein interaction networks and gene coexpression networks have unique node labels. More efficient algorithms have been proposed for the analysis of graphs with unique labels. The Mule algorithm [28, 29] proposed novel enumera-

[1]The material in this chapter was co-authored by Saeed Salem and College of Engineering and Architecture. Eihab El Radie had primary responsibility for collecting samples in the field and for interviewing users of the test system. Eihab El Radie was the primary developer of the conclusions that are advanced here. Eihab El Radie also drafted and revised all versions of this chapter. Saeed Salem served as proofreader and checked the math in the statistical analysis conducted by Eihab El Radie.

tion approaches for all maximal and closed frequent subgraphs, respectively. The Mule algorithm proposed an efficient enumeration tree in which each frequent subgraph appears exactly once, thus eliminating the need for duplicate checking. The algorithm is a depth first search that allows for deciding whether a subgraph is maximally frequent.

On large graph such as gene coexpression networks that have millions of coexpression links, the Mule algorithm takes takes hours to days depending on the minimum support threshold. With the availability of multi-core processors for personal machines, it is important to design parallel implementation of frequent subgraph mining algorithms. In this paper, we propose a multi-threaded implementation of the Mule algorithm [28].

In the next section, we discuss the problem description and define important terms. In Section 3, we present the multithreaded algorithm and discuss it in detail. Finally, in Section 4, we present the experiments we conducted on real gene co-expressions networks with biological analysis as well as the experiments performed on synthetic data.

3.2. Problem Description

We first introduce some notations and terms that are used throughout the chapter.

Let $\mathcal{G} = \{G_1, G_2, \cdots, G_n\}$ denote a set of n undirected graphs, where an undirected graph $G_i = (V, E_i)$ is a tuple where $V = \{v_1, v_1, \cdots, v_k\}$ is the set of vertices, and $E \subseteq V \times V$ is the set of edges.

Subgraph A graph $G_s = (V_s, E_s)$ is a *subgraph* of $G = (V, E)$, denoted as $G_s \subseteq G$, if and only if $V_s \subseteq V$ and $E_s \subseteq E$.

Frequent Subgraph Given a set of graphs $\mathcal{G}$ and user-specified support threshold S_{min}, a graph G_s is called frequent if it is a subgraph of at least S_{min} graphs. Let $sup(\mathcal{G}, G_s) = \{G_i | G_s \subseteq G_i \; and \; G_i \in \mathcal{G}\}$ denote the supporting graphs of G_s. A subgraph G_s is *frequent* if $|sup(\mathcal{G}, G_s)| \geq S_{min}$.

Maximal Frequent Subgraph let $\mathcal{F}$ denote the set of all frequent subgraphs. A frequent subgraph $X \in \mathcal{F}$ is called *maximal* if it has no frequent supersets. Let $\mathcal{M}$ be the set of all maximal frequent subgraphs, given as

$$\mathcal{M} = \{X \mid X \in \mathcal{F} \; and \; \nexists Y \supset X \; and \; Y \in \mathcal{F}\}$$

We mine for maximal frequent subgraphs as connectivity is an important feature that captures the relationship between the vertices and allows for eliminating mining disconnected combinations.

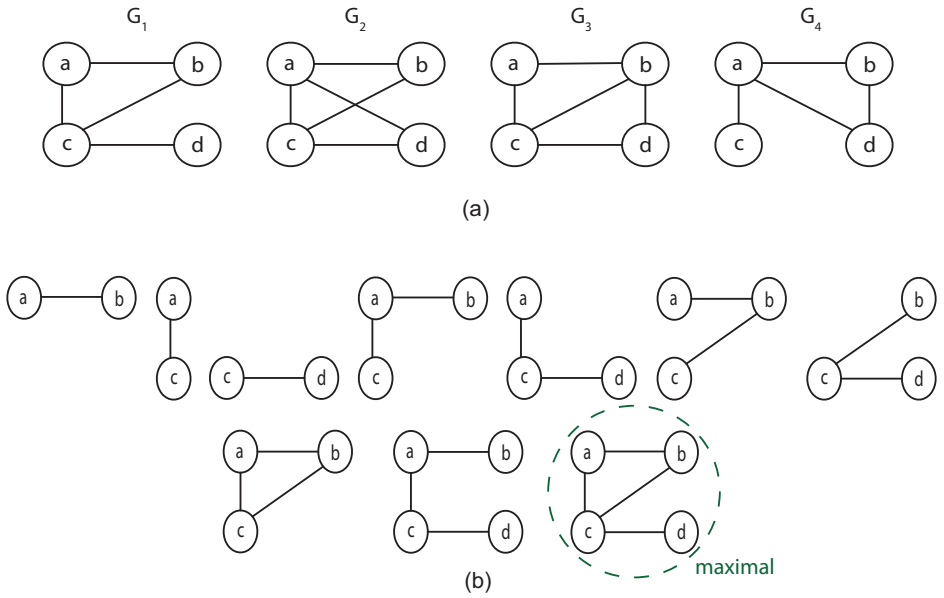


Figure 3.1. Example of frequent graph mining. (a) Graph dataset. (b) Frequent subgraphs of (a) with $S_{min} = 3$

Example Figure 3.1(a) shows a graph dataset with four graphs. For minimum support of 3, $S_{min} = 3$, there are ten frequent subgraphs, shown in Figure 3.1(b). In the third row, the subgraph with three edges $\{(a, b), (a, c), (b, c)\}$ is a frequent subgraph as it appears in G_1, G_2, and G_3. The set of all maximal frequent subgraphs $\mathcal{M}$ contains only one pattern that has four edges $\{a-b, a-c, b-c, c-d\}$ (enclosed in dashed circle). All the other frequent subgraphs are subgraphs of this maximal subgraph.

3.3. Algorithm

In this section we discuss a parallel approach, for mining maximal frequent subgraphs. The algorithm begins by constructing an enumeration tree where each edge in the tree is a frequent subgraph. Figure 3.2(a) shows a sample input graph and (b) shows its enumeration tree. After populating level one, a frequent subgraph is extended by adding one frequent edge from the neighboring candidate edges of the subgraph. The figure also shows all the frequent subgraphs, including the set of maximal frequent subgraphs which are in rectangular boxes and patterns that are not frequent are marked with crosses.

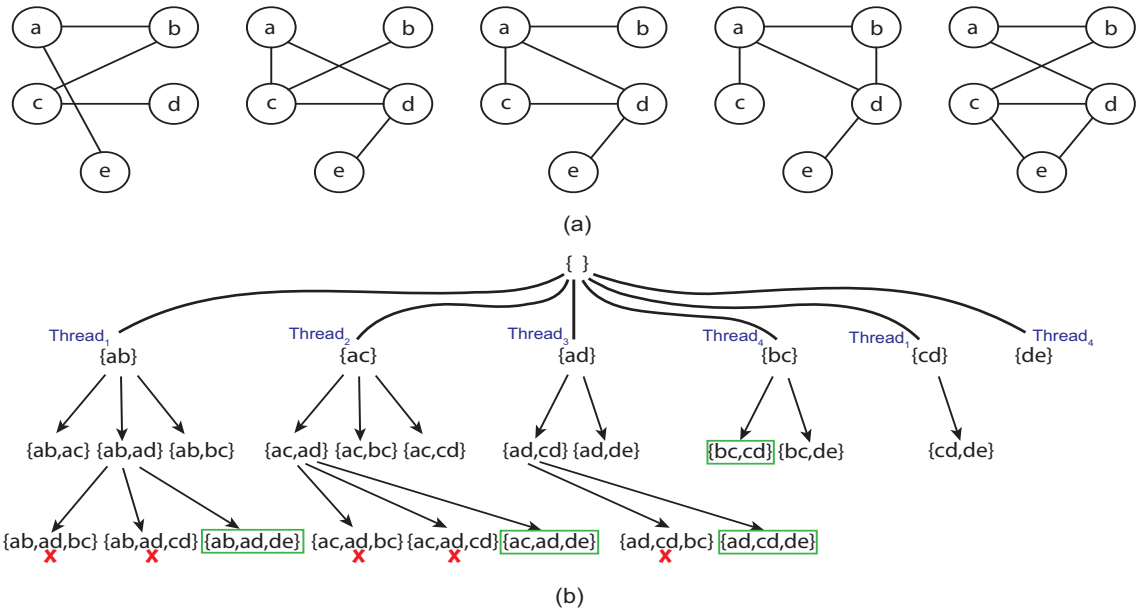


Figure 3.2. Maximal frequent subgraph mining works in parallel. (a) The input graphs database. (b) The resulting enumeration tree of maximal frequent subgraphs. Maximal patterns in boxes

The proposed algorithm utilizes a depth-first search enumeration technique based on backtracking [14] to mine maximal frequent subgraphs. This technique guarantees a unique child parent relationship, thus eliminating the need to check whether a subgraph has been visited before. This is a great improvement over the baseline technique where the repeated checking in an exponentially growing list of subgraphs consumes a tremendous amount of time.

We are interested in connected subgraphs, so we are only considering the connected subgraph. While extending a search node, we only add edges that are connected to the current subgraph. To ensure that we do not visit the same frequent subgraph again we maintain lists of visited and candidate edges.

The pseudocode for the proposed multithreading approach is shown in Algorithm 1. It takes as input: graph dataset, $\mathcal{G}$, a minimum support threshold, $minSup$, and a number of threads, $threadCount$. The algorithm begins by spawning the threads and each thread has his local set of maximal frequent patterns, $M_{t.id}$.

Each thread takes one of the unexplored frequent edges as a subgraph from level one (line 8). For each subgraph, we maintain a set of candidate edges, C_k, which contains the set of edges that can extend the current subgraph. Also we maintain a set of visited edges, D, that contains all the edges that have been visited in the current tree branch that is rooted at the current frequent

Algorithm 1 Pseudo-code for a parallel (Multithreaded) algorithm for Maximal Frequent Subgraph Detection

Input:
$\mathcal{G}$: Graph dataset
$minSup$: Minimum support of seeds
$threadCount$: User-specified number of threads
Output:
$\mathcal{M}$: Set of maximal frequent subgraphs
$M = \{M_1, M_2, ..., M_k\}$

1: $L1$: Frequent edges in $\mathcal{G}$
2: $threads[] = spawn\_threads(threadCount)$
3: $start\_all\_threads(threads[], ThreadStart)$
4: $join\_all\_threads(threads[])$
5: **function** ThreadStart
6: **while** there are unexplored frequent edges in L_1 **do**
7: Ensuring mutual exclusion, choose e_k
8: $G_k \leftarrow \{e_k\}$
9: $C_k \leftarrow \{e_j \in Neighbors(e_k) \mid e_j > e_k\}$
10: $D \leftarrow \{e_1, e_2, ..., e_k\}$
11: $execute_thread($MuleExtension$(M_{t.id}, G_k, C_k, D))$
12: **end while**
13: **end function**

14: **function** MuleExtension$(M_{t.id}, G_k, C_k, D)$
// C_k: the Set of candidate edges of G_k
//D: the set of previously visited edges
15: $isMaximal \leftarrow true$
16: **for** all edges $e_i \in C_k$ **do**
17: $D \leftarrow D \cup \{e_i\}$
18: $G_{k+1} \leftarrow G_k \cup \{e_i\}$
19: **if** G_{k+1} is frequent **then**
20: $isMaximal \leftarrow false$
21: $C_{k+1} \leftarrow (C_k \cup Neighbors(e_i)) \setminus D$
22: MuleExtension$(M_{t.id}, G_{k+1}, C_{k+1}, D)$
23: **end if**
24: **end for**
25: **if** $isMaximal$ **then**
26: **if** G_k *has no superset in* $M_{t.id}$ **then**
27: $M_{t.id} \leftarrow M_{t.id} \cup G_k$
28: **end if**
29: **end if**
30: **end function**
Exclude redundancies while grouping the sets in M in one Set $\mathcal{M}$

edge at level one. For each frequent subgraph, we extend the subgraph with one of candidate edges (line 17). We update the set of visited edges (line 18) and if the new subgraph is frequent we update the list of candidate edges for the new subgraph and recursively call MuleExtension. While extending the current subgraph with an edge, e_i, the new set of candidates will be the set of candidate edges of the current subgraph and the set of neighbors of the new added edge, minus the previously visited edges (line 22).

If the current subgraph cannot be extended with any of the edges in the candidate set, the subgraph is potentially maximal, and we check that there is no supergraph in the local set of maximal frequent patterns, $M_{t.id}$, (lines 26-30).

We keep a local list of maximal frequent patterns to avoid maintaining a shared list of all maximal frequent patterns. This eliminates the need to ensure mutual exclusion to the global list. However, these local maximal frequent patterns lists might have redundancy. Therefore, after all the threads are done their work, we merge the lists and remove any pattern that is not maximal.

Example In Figure 3.2(a) we have five graphs. Figure 3.2(b) shows the enumeration tree for mining subgraphs that appear in at least three graphs. Each thread will handle invoking the procedure, MuleExtension, for every edge in $L1 = \{a-b, a-c, a-d, b-c, c-d, d-e\}$ since these are the only frequent edges in the collection of input graphs. For instance, Neighbors$(a-c) = \{a-b, a-d, b-c, c-d\}$ and the set of candidate edges, C_k is $\{a-d, b-c, c-d\}$ and D contains $\{a-b, a-c\}$. When extending edge $a-d$, Neighbors$(a-d) = \{a-b, a-c, c-d, d-e\}$ and the set of candidate edges, C_k is $\{c-d, d-e\}$ and D contains $\{a-b, a-c, a-d\}$.

3.4. Experimental Results

In this section, we conduct experiments to evaluate the performance of the proposed algorithm. The algorithm is implemented in C++ and experiments were conducted on a system with an Intel Xeon (3.1GHz) processor, 16 cores, 16 GB RAM and Ubuntu operating system.

3.4.1. Datasets

Experiments are conducted on two different datasets with varying parameters. We have used real coexpression networks, and synthetic data for runtime comparison purposes.

1. Synthetic data

To investigate how different dataset characteristics impact the performance and speedup of the algorithm, we generated synthetic datasets with different parameters.

Generation tool: We implemented our own data generator for synthetic datasets, which generate synthetic datasets as follows. The data generator creates n graphs $\{G_1, G_2, ..., G_n\}$ which has the set of vertices U such that for each graph $\{G_i\}_{i=1}^{n}$, $V(G_i) = U$ and $E(G_i) = \emptyset$. The next step is, given the number of maximal frequent patterns, $\mathcal{M}$, and the user-specified parameter, $PatternSize$, the data generator randomly generates $\mathcal{M}$ maximal frequent patterns and embed each one into a set of $minSup$ randomly selected graphs.

In the experiment reported in this section, the values of the parameters were as follows:

1. The first sets of datasets used are $PatternSize = 23$ edges (10 nodes) and $\mathcal{M}$ from $\{10, 20, 30, 40, 50\}$.

2. The second sets of datasets used are of the combinations of $PatternSize = \{16, 18, 19, 21, 23\}$ and $\mathcal{M} = 100$.

2. GTEx

We use the GTEx tissue-specific RNA-seq expression data generated for more than thirty distinct human tissues [27]. The goal in this dataset is to learn tissue-specific gene regulatory networks for 35 human tissues using the tissue-specific gene expression data, where two genes are linked if their expression levels are correlated. This dataset is inferred from 35 tissues, $10,000$ genes, and 5 million edges. When we prune edges that appear in a small number of networks, the number of edges decrease dramatically. There are $771,676$ and $330,101$ edges that appear in at least 3 and 5 networks, respectively. For the final dataset, we pruned infrequent edges that appear in less than 10 networks, leaving $55,458$ frequent edges.

3.4.2. Runtime

In this section, we compare the runtime of our algorithm with varying number of threads. Figure 3.3 and Figure 3.4 show the speedup on the first synthetic datasets. The speedup for 16 threads ranged from 10 to 14. For 32 threads, the speedup was 20 when the number of embedded patterns was 50.

For the second synthetic dataset where we varied the size of embedded pattern, the running time speedup is show in Figure 3.4. For instance, the execution of single thread takes approximately 5 hours for finding the maximal patterns with $PatternSize = 23$, while the parallel execution of the same experiment with 32 threads takes roughly 14 minutes, *speedup* of more than 21.

A very important aspect is that the running time for the algorithm depends on the depth of the search branches of the enumeration tree. Since the embedded patterns were of the same size, load balancing was fair among the threads on the synthetic datasets.

For GTEx dataset, the speedup changes significantly as the number of threads increase. In Figure 3.5 we noticed that the speedup increases when we use lower S_{min} constantly while

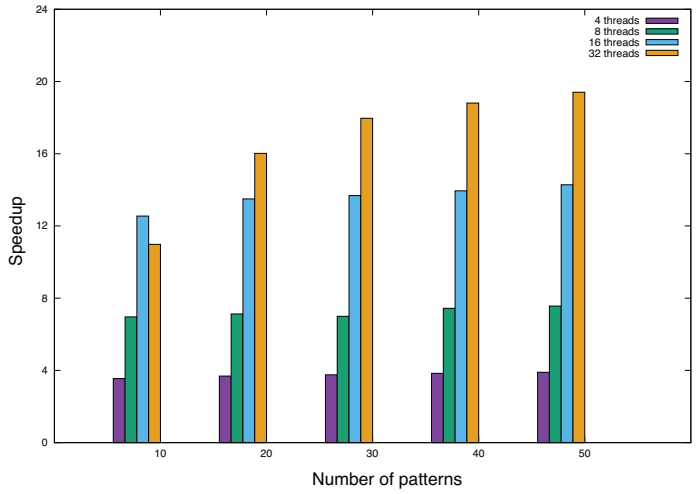


Figure 3.3. Synthetic data: Speed on varying number of patterns

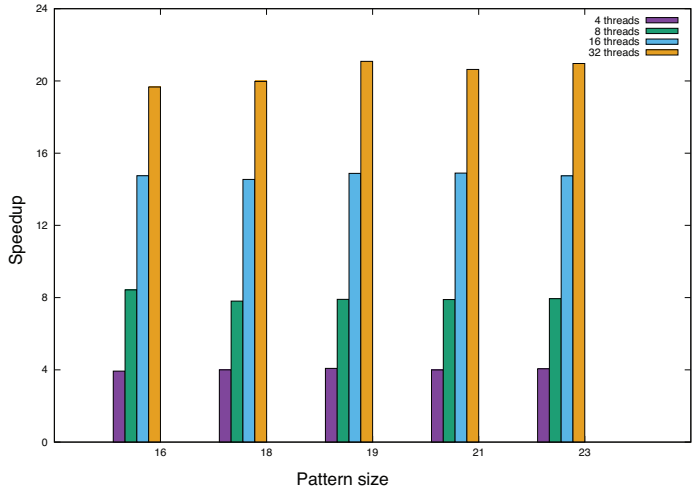


Figure 3.4. Synthetic data: Speed on varying pattern size

increasing number of threads as long as the number of threads is not much larger than the number of cores in the CPU.

For example, for finding the maximal patterns with $S_{min} = 17$, the execution of the algorithm using single thread takes approximately 51 minutes, but the parallel execution using the same characteristics but with 16 threads takes roughly 3 minutes. For higher support ($S_{min} = 19$), the algorithm takes less than 1 minute and the speedup is not high since most of the time is spent in reading the dataset. Our experiments with low minimum support, $S_{min} \leq 16$, took more than 5 days with single thread execution and hence we were not able to determine results for lower support.

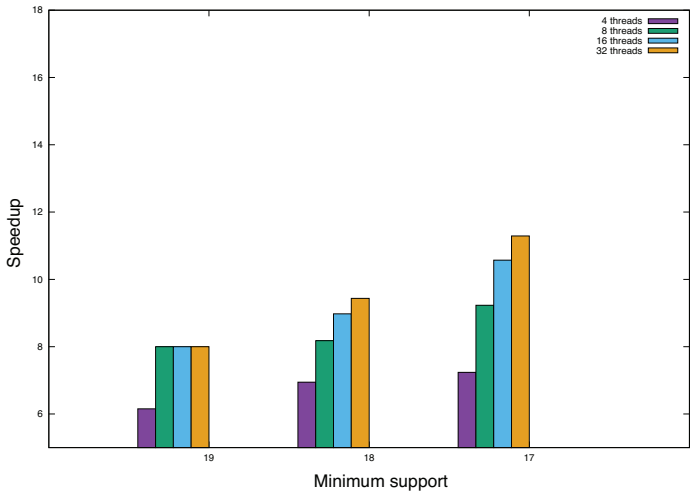


Figure 3.5. Real data: Speedup before fixing slow worker problem

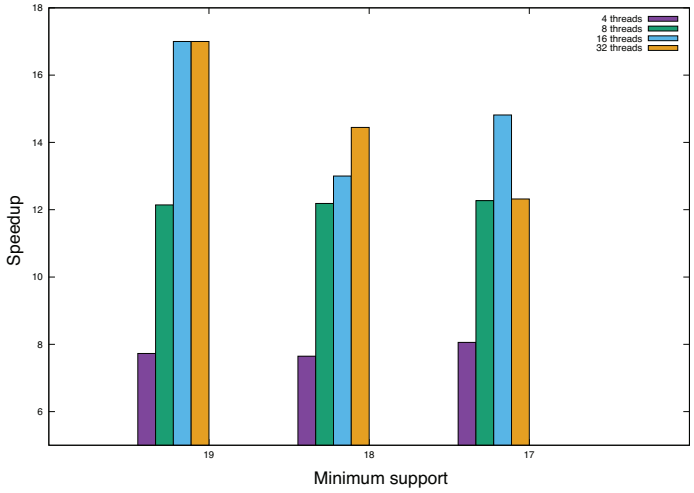


Figure 3.6. Real data: Speedup after fixing slow worker problem

One of the problems which we faced using multithreading on the GTEx dataset was the ***slow worker*** problem where one thread handling large branch in the enumeration tree takes too long, while all the other threads have finished their tasks, which increase the overall runtime. We fixed this problem by checking if any thread is idle, we give part of the branch the thread is working on to another idle thread. Therefore, no idle threads were present while others are working at the same time. Fixing the slow workers improved the speedup of the algorithm as shown in Figure 3.5 and Figure 3.6.

3.4.3. Biological Analysis

To assess the biological significance of the reported patterns, we performed an enrichment analysis of these gene sets. In annotation databases, genes are annotated with multiple terms and the same term can annotate many genes. If at least one biological annotation is overrepresented in the reported subgraph's genes, the subgraph is marked as enriched.

Over-representation analysis For a given annotation term, let N be the the number of the genes in the database, K denote the number of genes in the database that are annotated, let n be the number of genes in the subgraphs, and k is the number of genes of the subgraph with the annotation.

The probability of getting k or more annotated genes if we randomly sample n genes can be calculated by the cumulative hypergeometric distribution as following:

$$P(X \geq k) = 1 - \sum_{i=1}^{k-1} \frac{\binom{K}{i}\binom{N-K}{n-i}}{\binom{N}{n}}$$

We performed biological enrichment analysis of the reported patterns for the GTEx dataset to assess the biological significance. We checked for enrichemnt (over-representation) of Gene Ontology (GO) biological process terms [2], KEGG (Kyoto Encyclopedia of Genes and Genomes) pathways [23, 25, 24], and Gene-Disease association (DisGeNET) [37]. We used the DAVID functional annotation tool for biological enrichment [17] .

Table 3.1. Enrichment analysis of maximal frequent subgraphs reported by the first algorithm.

S_{min}	N	$\overline{N}$	$\overline{Density}$	$GO\%$	$KEGG\%$	$DisGeNET\%$
17	6598	12.03	0.24	91.7	31.4	53.9
18	4116	10.95	0.25	92	27.7	55.9
19	2580	9.99	0.27	92	25.4	57.2
20	1701	9.01	0.29	91.8	27.3	58.6

Table 3.1 presents the topological properties and the enrichment percentage for varying minimum support. Topological properties of the reported patterns show that as we decrease S_{min}, the number of patterns and the average size of the patterns increase while the average density decrease.

Table 3.1 shows that the GO biological processes enrichment is above 91% in all conducted experiments, which indicates that the reported patterns are biologically relevant. In all the experiments, over 53% of all the reported patterns are enriched with Gene-Disease association

(DisGeNET), which indicated a strong association between the reported patterns and diseases. Moreover, the percentages of enriched patterns with KEGG pathways are shown in Table 3.1.

Table 3.2. Most enriched GO, DO Terms, and KEGG Pathways in the reported patterns of the first algorithm.

Top 5 GO, DO biological process terms	
GO:0006614	SRP-dependent cotranslational protein targeting to membrane
GO:0000184	nuclear-transcribed mRNA catabolic process, nonsense-mediated decay
GO:0044033	multi-organism metabolic process
GO:0042254	ribosome biogenesis
GO:0006613	cotranslational protein targeting to membrane
Top 5 KEGG Pathways	
hsa03030	DNA replication
hsa04931	Insulin resistance
hsa05033	Nicotine addiction
hsa00120	Primary bile acid biosynthesis
hsa05032	Morphine addiction
Top 5 Diseases	
DOID:4556	Lung large cell carcinoma
DOID:446	Hyperaldosteronism
DOID:5577	Gastrinoma
DOID:12678	Hypercalcemia
DOID:3852	Peutz-Jeghers syndrome

Some GO biological processes, KEGG pathways, and Disease Ontology terms are enriched in the genes of more than one pattern. Table 3.2 shows the top 5 GO biological processes, KEGG pathways, as well as top 5 diseases that were enriched in the most in the reported patterns.

Figure 3.7 and Figure 3.9 show two examples of two reported patterns that are enriched with GO terms and KEGG pathways. The subgraph in Figure 3.7 appears in 17 out of the 35 graphs in the dataset of real coexpression networks, while the subgraph in Figure 3.9 appears in 18. The edge-occurrence heatmaps of the two reported patterns are shown in Figure 3.8 and Figure 3.10.

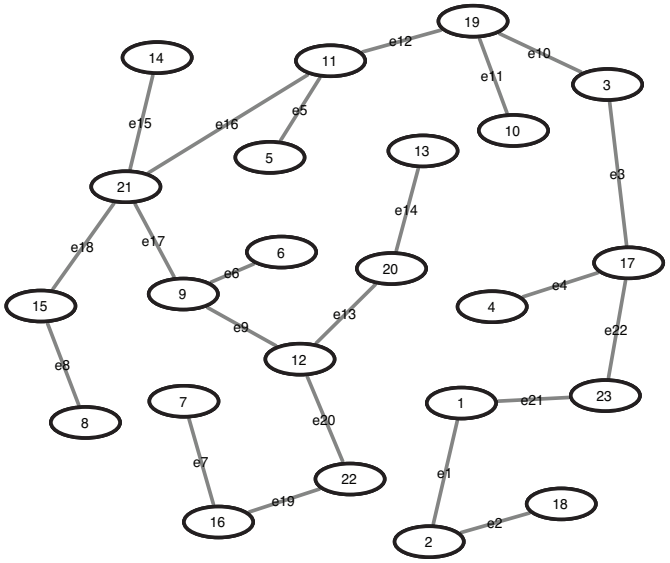


Figure 3.7. Example 1: maximal frequent subgraph. $S_{min} = 18$.

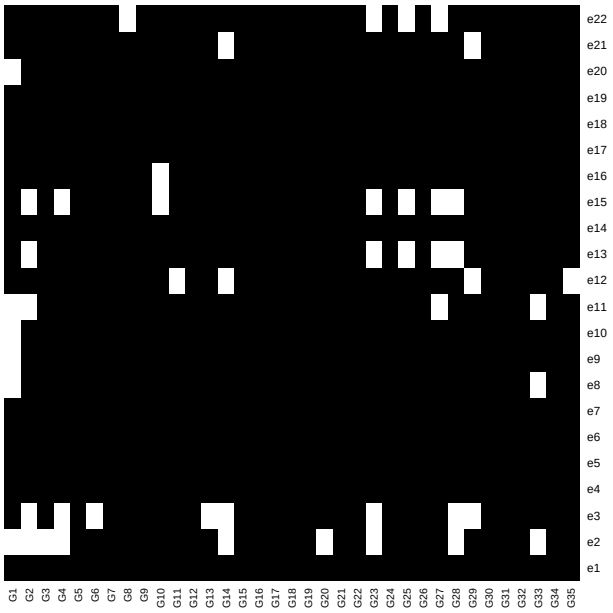


Figure 3.8. Edge-occurrence heatmap of Example 1.

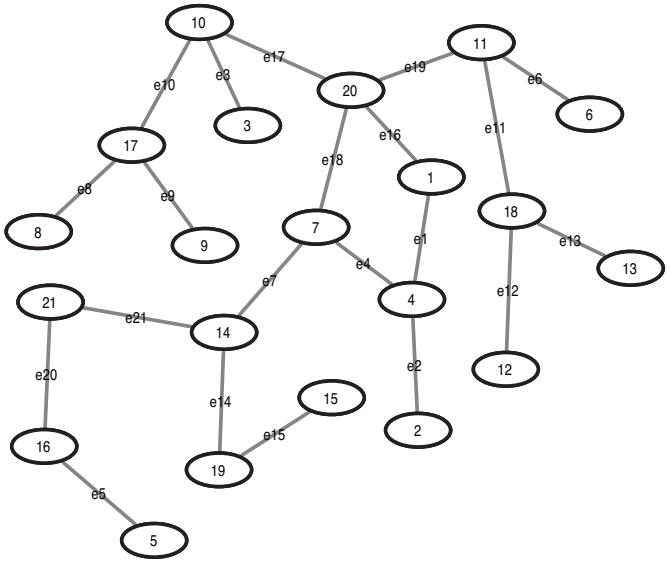


Figure 3.9. Example 2: maximal frequent subgraph. $S_{min} = 18$.

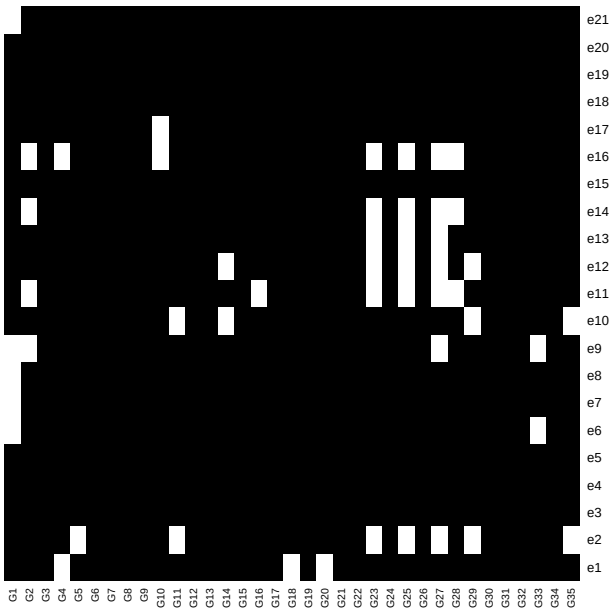


Figure 3.10. Edge-occurrence heatmap of Example 2.

4. MINING QUASI FREQUENT COEXPRESSION SUBNETWORKS[2]

4.1. Introduction

Recently, research has focused on integrating multiple gene expression datasets for identifying functional modules. Genes that show correlated expression profiles in multiple experiments have been proposed for module discovery and functional annotation [32, 16]. Integrative analysis of multiple omic datasets (e.g., interaction networks and gene expression) has the potential to elucidate the intricate interactions involved in biological processes, and has been employed for functional annotation [32], active module discovery [20], and biomarker discovery [8]. Efficient algorithms for discovering interesting patterns from coexpression networks have been proposed.

One of the early work for integrating multiple gene expression datasets was proposed in Lee et al. [32]. The approach extracts clusters from a summary network formed by the coexpression links that appear in at least a minimum number of coexpression networks. Lee et al. applied their approach on mRNA coexpression based on 60 large human data sets containing a total of 3924 microarraysm. In this work, they proved that the confirmed positive correlations between genes in the dataset were much more common than confirmed negative correlations. However, this approach suffers from false positive modules that appear highly connected in the summary network and not connected in the individual coexpression networks. To solve the problem of false positive modules, Hu et al. [16] proposed an approach to further partition the clusters extracted from the summary graph by employing the edge-occurrence similarity. However, false positive modules can still exist if the similarity threshold is not well chosen.

Other methods have mined frequent pattern from the edge occurrence in the graphs and then partitioned these pattern into connected subnetworks. Huang et al. [18] employed simulated

[2]The material in this chapter was co-authored by Saeed Salem and College of Engineering and Architecture. Eihab El Radie had primary responsibility for collecting samples in the field and for interviewing users of the test system. Eihab El Radie was the primary developer of the conclusions that are advanced here. Eihab El Radie also drafted and revised all versions of this chapter. Saeed Salem served as proofreader and checked the math in the statistical analysis conducted by Eihab El Radie.

annealing to mine semi-frequent edgesets that frequently appear in the coexpression networks. These edgesets are not necessarily connected and thus a second phase is to partition these edgesets to highly connected subnetworks. Salem and Ozcaglar [40] proposed an approach that mines maximal frequent edgesets, complete biclusters, and extracts highly-connected subgraphs (k-cliques and percolated k-cliques) from the induced subgraph for each edgeset.

Graph clustering approaches have been proposed for integrating multiple coexpression networks. Salem and Ozcaglar [41] proposed an approach that combines the summary graph information with edge recurrence. The approach combines both edge similarity in the summary graph with the edge similarity based on the common occurrence of the two edges in the individual graphs. Modules are then extracted from the combined edge-edge similarity graphs. An extension of [41] was proposed in Salem [39] where edge-edge topological similarities for each of the coexpression networks are combined in a summary graph from which modules are extracted.

In this work, we propose an algorithm that mines subgraphs that quasi 'appear' in at least a number of graphs. In the rest of the paper, the organization is done as follows. In Sections 2 and 3, we discuss the problem and we present our algorithm, respectively. Furthermore, in Section 4, we present experimental results on real gene coexpression networks.

4.2. Proposed Approach

We first introduce some notations and terms that are used throughout the thesis. Let $D = \{G_i\}_{i=1}^{n}$ denotes a dataset of $\mathbf{n}$ *undirected* graphs, where for each graph $G_i = (V, E)$, $V = \{v_1, ..., v_k\}$ is the set of vertices, and $E \subseteq V \times V$ is the set of edges. A graph $G_s = (V_s, E_s)$ is a *subgraph* of $G = (V, E)$, denoted as $G_s \subseteq G$, if and only if $V_s \subseteq V$ and $Es \subseteq E$.

Frequent Subgraph Given a minimum support threshold S_{min}, a graph G_s is called a frequent subgraph in D if it is a subgraph of at least S_{min} graphs in D. Let $sup(D, G_s) = \{G_i | G_s \subseteq G_i \; and \; G_i \in D\}$ denote the supporting graphs of G_s. A subgraph G_s is a frequent subgraph if $|sup(D, G_s)| \geq S_{min}$.

Mining for frequent subgraphs requires all the edges of the subgraph to be present in each of the supporting graphs. Some of the edges might be missing due to noise in data generation, thus leading to missing some important patterns. We propose to allow some missing edges while counting the support of a subgraph. A subgraph G_s has $t-$appearance in a graph if $t \times |E(G_s)|$

edges are present in the graph. The same subgraph can multiple t-appearance in different graphs in the dataset.

Quasi Frequent Subgraph Given a minimum support threshold S_{min}, and density threshold, t, a graph $G_s = (V_s, E_s)$ is called a quasi frequent subgraph in D if the subgraph has $t-$appearance in at least S_{min} graphs in D.

Let $t\text{-}sup(D, G_s) = \{G_i | \, |E(G_s) \cap E(G_i)| \geq t \times E(G_i) \; and \; G_i \in D\}$, denote the set of graphs in which the subgraph has a t-appearance. A subgraph is quasi frequent if $t\text{-}sup(D, G_s) >= S_{min}$.

The definition of quasi subgraph does not impose restrictions on the number of graphs in which each edge of the subgraph should appear. An edge can still appear in a small number of graphs and yet be part of a quasi frequent subgraph. We further require that each edge of subgraph appear in at least t of the supporting graphs.

4.3. Algorithm

We introduce a two-step approach for mining a representative set of quasi frequent subgraphs.

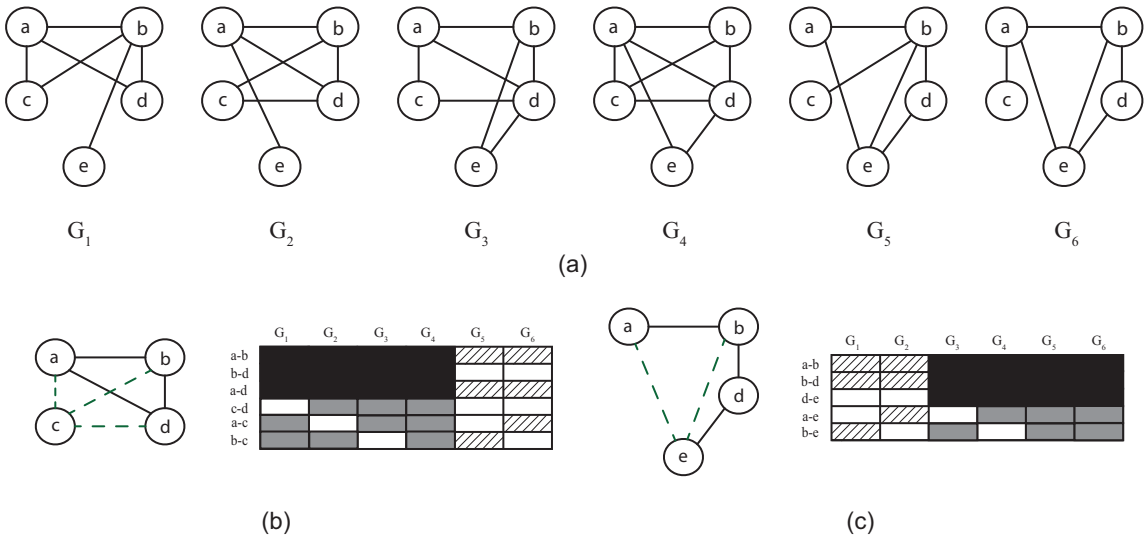


Figure 4.1. (a) Graph dataset with six graphs. (b) Extension of seed {(a,b),(a,d),(b,d)} with three edges; (c) Extension of seed {(a,b),(b,d),(d,e)} with two edges. Parameters used: $S_{min} = 3$ and $t = 0.7$

Seed Generation Given a graph dataset, D, minimum support, S_{min} and size of seed subgraph, *seedSize*, we employ the MULE algorithm for mining all frequent subgraphs with size equals *seedSize* edges. The algorithm enumerates frequent subgraphs without redundancy check-

ing. When the size of the subgraph equals $seedSize$, the algorithm calls the seed extension procedure. The seed generation is shown in Algorithm 2. The algorithm builds an enumeration tree by starting from each frequent edge (subgraph) and it expends the subgraph by trying to add one of its neighboring edges. If the newly formed subgraph (line 7) is frequent, then the procedure is recursively called with the new subgraph as the starting pattern (line 10). Two sets, V_s and C_k are maintained to avoid visiting the same subgraph patterns multiple times.

Figure 4.1 (a) shows a graph dataset with six graphs, with minimum support equals 3, (b) and (c) show examples of two seeds, $\{(a,b),(a,d),(b,d)\}$ and $\{(a,b),(b,d),(d,e)\}$ (shown with solid lines). Each of these seeds appear in four graphs.

Seed Extension The greedy pattern growth extends each seed, G_{si}, with the neighboring edge that maximizes the number of occurrences in the set of supporting graphs of the original seed. For each seed, we have the associated supporting subgraphs in which the seed completely appear, i.e., $sup(D, G_{si})$. To ensure that the newly formed subgraph is a quasi frequent, we require each of the supporting graphs of the original seed to have at least t of the edges of the subgraph. Moreover, we require the newly added edge to appear in at least t the supporting graphs of the original seed. If the current subgraph cannot be extended with a valid edge, then the current subgraph is a maximal quasi frequent subgraph. Note that a seed subgraph is a quasi frequent subgraph.

Example Figure 4.1 (b) and (c) show two seeds that have been extended. Each of these seeds have four supporting graphs. For the seed in (b), we repeatedly add add the following edges: $\{(c,d),(a,c),(b,c)\}$. Each of the four original graphs have at least 0.7 of the edges in the subgraph. Moreover, each added edge appears in three of the supporting graphs. After adding these three edges, no more edges can be added without violating the conditions. Therefore, the subgraph formed in (b) is a maximal subgraph. Figure 4.1 (c) shows the extension of the seed with only two edges.

4.4. Experimental Results

In this section, we conducted experiments on real coexpression networks to evaluate the performance of the proposed algorithm. The algorithm is implemented in C++ and we ran the experiments on a system with an Intel Xeon (3.1GHz) processor, 8 GB RAM and Ubuntu operating system.

Algorithm 2 Pseudo-code for Seed Generation and Greedy Pattern Growth

Input:
D: Graph dataset
$seedSize$: seed size
$minSup$: minimum support of seeds
t: user-specified threshold of the relaxation
Output:
$\mathcal{F}$: Set of reported subgraphs
Call MineFreqGraphsfor each frequent edge

$//C_k$: the set of candidate edges of G_k
$//V_s$: the set of previously visited edges

function MineFreqGraphs($\mathcal{F}, G_k, C_k, V_s$)
  if $\mid G_k \mid == seedSize$ **then**
    GreedyPatternGrowth(F, G_k)
  end if
  for all edges $e_i \in C_k$ **do**
    $V_s \leftarrow V_s \cup \{e_i\}$
    $G_{k+1} \leftarrow G_k \cup \{e_i\}$
    if G_{k+1} is frequent **then**
      $C_{k+1} \leftarrow (C_k \cup Neighbors(e_i)) \setminus V_s$
      MineFreqGraphs($\mathcal{F}, G_{k+1}, C_{k+1}, V_s$)
      return
    end if
  end for
end function

function GreedyPatternGrowth($\mathcal{F}, G_k$)
  $e_{idx} = -1$
  $maxDensity = -1$
  $Extend = false$
  for all edges $e_i \in Neighbors(G_k)$ **do**
    $G_{k+1} \leftarrow G_k \cup \{e_i\}$
    //Check if the added edge meets the conditions.
    $currentDensity = calculateDensity(G_{k+1})$
    if $currentDensity > maxDensity$ **then**
      $Extend = true$
      $e_{idx} = e_i$
      $maxDensity = currentDensity$
    end if
  end for
  $G_{k+1} \leftarrow G_k \cup \{e_{idx}\}$
  if $Extend == true$ **then**
    GreedyPatternGrowth($\mathcal{F}, G_{k+1}$)
  else
    $\mathcal{F} \leftarrow \mathcal{F} \cup G_k$
  end if
end function

4.4.1. Dataset

We use the GTEx tissue-specific RNA-seq expression data generated for 35 distinct human tissues [27]. This dataset is inferred from 35 gene expression dataset and has over 5 million coexpression links between $10,000$ genes.

4.4.2. Topological Analysis

We analyzed the subgraphs reported by our algorithm on the GTEx dataset. Table 4.1 presents the topological properties of the reported patterns for varying minimum support, S_{min} and density t, thresholds, while maintaining the value to be $seedSize = 3$. Table 4.1 shows that most of the patterns are quite dense with density more than 0.4. As the density, t, is decreased to allow for more edges to be added to the seed, we get larger patterns as evident in the average size of the reported pattern. However, we get also less number of patterns as some patterns will be included in other maximal patterns and we remove redundancy from the final reported patterns.

The algorithm is extremely fast even for support as low as 12. This is important considering that mining all frequent subgraphs takes hours to finish. We ran the MULE algorithm [28] on the same minimum support and it took more than 160 hours without getting the results.

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

S_{min}	t	N	$\overline{N}$	$\overline{Density}$	$Runtime(sec)$
12	0.7	21,934	5.44	0.43	85
	0.8	25,866	4.05	0.46	69
	0.9	29,008	3.6	0.47	65
13	0.7	13,439	4.54	0.45	28
	0.8	15,626	3.86	0.46	25
	0.9	18,121	3.49	0.48	24
14	0.7	7,623	5.42	0.43	14
	0.8	9,108	4.24	0.46	13
	0.9	11,340	3.54	0.48	11
15	0.7	4,690	5.96	0.42	6
	0.8	5,569	4.28	0.45	6
	0.9	7,167	3.59	0.48	5
16	0.7	2,832	5.78	0.42	4
	0.8	3,563	4.31	0.45	3
	0.9	4,445	3.72	0.47	3
17	0.7	2,144	5.39	0.43	2
	0.8	2,423	4.44	0.45	2
	0.9	2,942	3.83	0.47	1
18	0.7	1,276	13.8	0.38	2
	0.8	1,539	4.75	0.43	1
	0.9	1,921	4.03	0.45	1
19	0.7	1,020	5.3	0.42	1
	0.8	1,213	4.58	0.44	1
	0.9	1,392	4	0.46	1
20	0.7	698	5.75	0.41	1
	0.8	842	4.82	0.42	<1
	0.9	962	4.14	0.45	<1

4.4.3. Biological Analysis

Similar to section 3.4.3, we performed biological enrichment analysis of the reported patterns for the GTEx dataset to assess the biological significance. We checked for enrichemnt (over-representation) of Gene Ontology (GO) biological process terms [2], KEGG (Kyoto Encyclopedia of Genes and Genomes) pathways [23, 25, 24], and Gene-Disease association (DisGeNET) [37]. We used the DAVID functional annotation tool for biological enrichment [17]. If a biological annotation is overrepresented in the reported subgraph's genes, the subgraph is marked as enriched.

To assess whether the coexpression subnetworks are enriched with physical protein-protein interactions (PPIs), we analyzed the enrichment of PPIs in the reported subnetworks. If PPIs are significantly overrepresented ($p - value <= 0.01$) in the reported subnetworks, we consider the subnetwork as enriched.

We used the BIOGRID protein-protein interaction network (version 3.4.152) that has $287,970$ interactions [7]. Almost 25% of the reported patterns are enriched with PPIs. The percentage could be higher given that current experimental high-throughput technologies have a high rate of missing interactions [48].

Table 4.2. Enrichment analysis of the reported patterns by the second algorithm with different ontology databases

S_{min}	t	N	$PPI\%$	$GO\%$	$KEGG\%$	$DisGeNET\%$
16	0.7	2832	21	75.8	62.8	52.1
	0.8	3563	25	76.7	62.4	52.8
	0.9	4445	24	76.1	58.9	50.2
17	0.7	2144	30	79.4	66.9	54
	0.8	2423	31	81.4	67.2	53.6
	0.9	2942	27	78.7	63.5	50.8
18	0.7	1276	24	80.5	64.3	57.2
	0.8	1539	29	82.6	67.1	50.6
	0.9	1921	27	81.3	63.9	51
19	0.7	1020	32	83	71.8	54.2
	0.8	1213	32	84.3	69.7	53.4
	0.9	1392	28	82.6	66.9	52.1
20	0.7	698	27	84.7	71.2	53.7
	0.8	842	30	85.2	70.4	53.4
	0.9	962	27	85.7	66.7	52.5

Table 4.2 provides the enrichment percentage of the reported patterns for varying minimum support and density thresholds using different ontology databases. The GO biological processes enrichment in above 75% in all the cases and increases as we increase the minimum support. This indicates that more frequent patterns are more likely to be biologically relevant. In all experiments, over 50% of all the reported patterns are enriched with Gene-Disease association (DisGeNET),

which indicated a strong association between the reported patterns and diseases. Moreover, the table shows the percentages of enriched patterns with protein-protein interactions patterns, KEGG pathways.

Table 4.3. Examples of PPI enrichment of 10 reported patterns by the second algorithm

Pattern	Pattern Size	Obs. PPI	p-value
ptrn1	130	88	1.64E-219
ptrn2	95	73	1.99E-189
ptrn3	91	63	1.26E-158
ptrn4	87	61	3.29E-154
ptrn5	89	61	3.40E-153
ptrn6	59	49	6.81E-131
ptrn7	60	49	3.71E-130
ptrn8	60	49	3.71E-130
ptrn9	61	49	1.88E-129
ptrn10	59	48	2.28E-127

Table 4.3 lists examples of 10 patterns that are enriched with PPIs. For example, $ptrn1$ has 130 edges, of which 88 are true PPIs.

Moreover, some Disease Ontology terms are enriched in the genes of more than one pattern. Table 4.4 shows the top 10 diseases that were enriched the most in the reported patterns.

Table 4.4. Top 10 diseases enriched in the reported patterns by the second algorithm.

Top 10 Diseases	
DOID:12449	aplastic anemia
DOID:11335	sarcoidosis
DOID:2916	hypersensitivity reaction type IV disease
DOID:16	integumentary system disease
DOID:2730	epidermolysis bullosa
DOID:2731	vesiculobullous skin disease
DOID:0060036	intrinsic cardiomyopathy
DOID:4960	bone marrow cancer
DOID:2355	anemia
DOID:13378	Kawasaki disease

Figure 4.2 and Figure 4.4 shows two reported patterns that are enriched with GO, DO terms, and PPIs. The edge-occurrence heatmaps of the two reported patterns are shown in Figure 4.3 and Figure 4.3. The parameters used in these examples are $S_{min} = 3$ and $t = 0.7$.

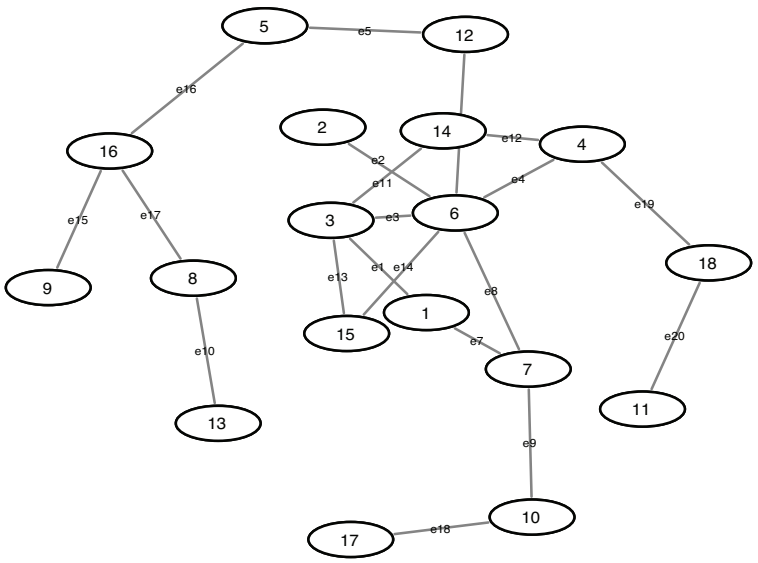


Figure 4.2. Example 1: maximal quasi frequent subgraph. $S_{min} = 16$ and $t = 0.7$

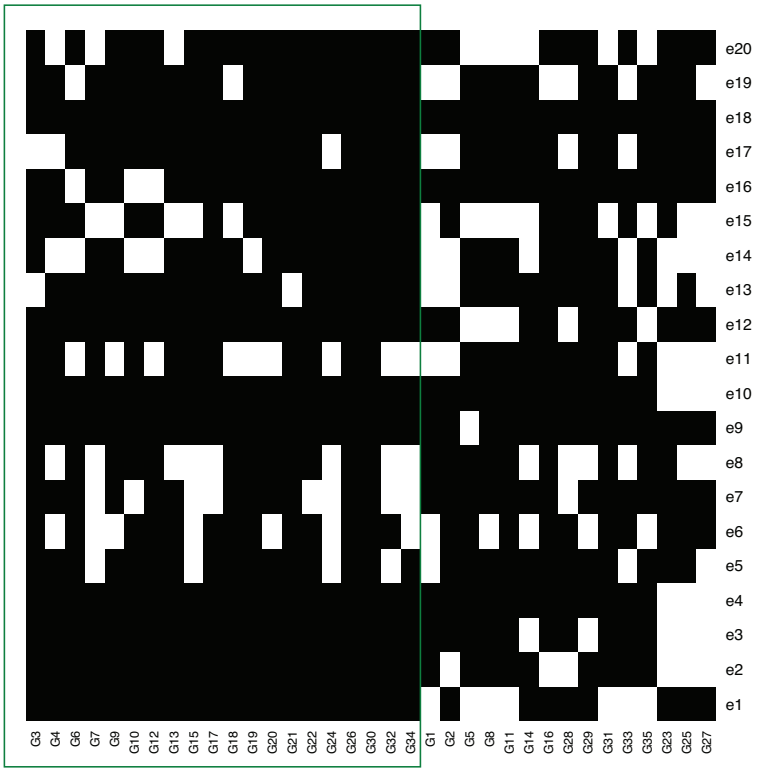


Figure 4.3. Edge-occurrence heatmap of Example 1.

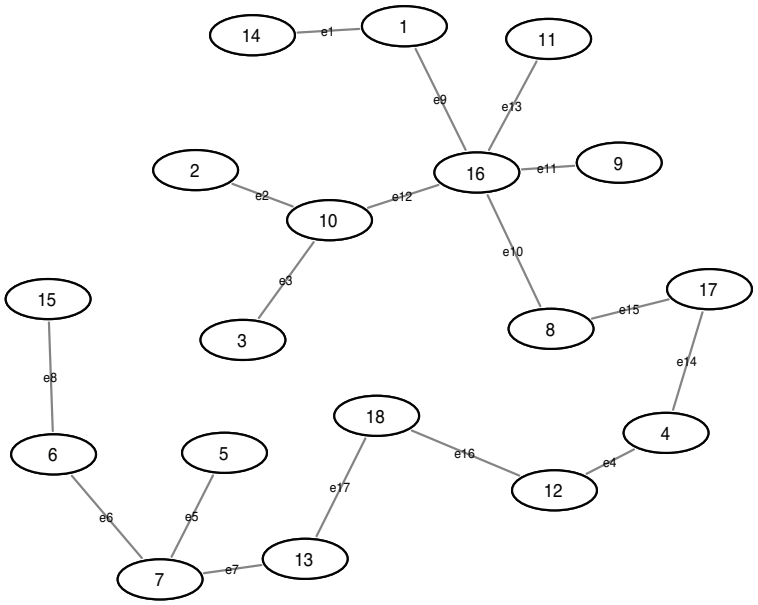


Figure 4.4. Example 2: maximal quasi frequent subgraph reported by the second algorithm. $S_{min} = 16$ and $t = 0.7$

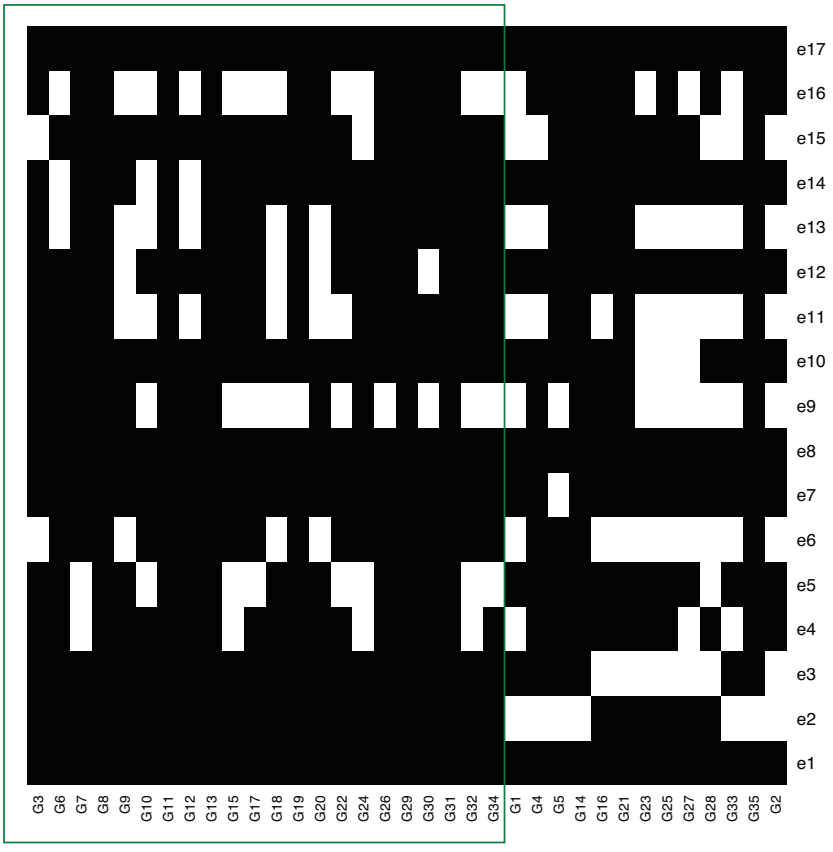


Figure 4.5. Edge-occurrence heatmap of Example 2.

5. CONCLUSION AND FUTURE WORK

In this thesis, we proposed algorithms for mining frequent subgraphs. In the first algorithm, we proposed a multithreaded algorithm for mining maximal frequent subgraphs from large graph datasets. We have demonstrated the effectiveness of the parallel implementation on synthetic and real datasets. On a real dataset where we have different frequent pattern sizes, threads that work on deeper branches take much longer to complete. Fixing the slow worker problem improves the speedup. Even though the first algorithm is relatively efficient, it took much time when using small support values and it does not allow relaxation while mining the subgraphs. To overcome those challenges we developed a second algorithm. In the second algorithm, we proposed a greedy approach for mining quasi frequent subgraphs. Experiments on synthetic data as well as real biological networks are done and we showed the effectiveness and efficiency the algorithms achieve and the big improvement in performance. Biological enrichment analysis of the reported patterns show that the patterns are biologically relevant and enriched with known biological processes and disease terms.

Future work can address improving the algorithms to produce frequent patterns with minimal overlapping among each other. In the two algorithms, the reported frequent patterns do not change dramatically over time. Therefore, the tree structure will likely help producing considerable overlap. By discovering a method to reduce the overlapping among the produced patterns, we will make a biological impact by reporting only the most important patterns.

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. *Proceedings of the 1994 International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, September 1994.

[2] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25–29, 2000.

[3] Sitaram Asur and Bernardo A Huberman. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE, 2010.

[4] Albert-Laszlo Barabasi and Zoltan N. Oltvai. Network biology: understanding the cell's functional organization. *Nat Rev Genet*, 5(2):101–113, 2004.

[5] Pavel Berkhin. *A Survey of Clustering Data Mining Techniques*, pages 25–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[6] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[7] Andrew Chatr-aryamontri, Bobby-Joe Breitkreutz, Sven Heinicke, Lorrie Boucher, Andrew Winter, Chris Stark, Julie Nixon, Lindsay Ramage, Nadine Kolas, Lara O'Donnell, et al. The biogrid interaction database: 2013 update. *Nucleic acids research*, 43(D1):D470–D478, 2015.

[8] Han-Yu Chuang, Eunjung Lee, Yu-Tsueng Liu, Doheon Lee, and Trey Ideker. Network-based classification of breast cancer metastasis. *Mol Syst Biol.*, 3:140, 2007.

[9] Charles E Cook, Mary Todd Bergman, Robert D Finn, Guy Cochrane, Ewan Birney, and Rolf Apweiler. The european bioinformatics institute in 2016: Data growth and integration. *Nucleic Acids Research*, 44(Database issue):D20–D26, 2016.

[10] Diane J. Cook and Lawrence B. Holder. Substructure discovery using minimum description length and background knowledge. *J. Artif. Int. Res.*, 1(1):231–255, 1994.

[11] Padraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers. *Multiple Classifier Systems*, 34:1–17, 2007.

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

[13] B. Goethals. Survey on frequent pattern mining. Technical report, Helsinki Institute for Information Technology, 2003.

[14] Karam Gouda and Mohammed Zaki. Efficiently mining maximal frequent itemsets. In *ICDM '01 Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 163–170. IEEE, 2001.

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

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

[17] Da Wei Huang, Brad T. Sherman, and Richard A. Lempicki. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Res.*, 37(1):1–13, 2009.

[18] Yu Huang, Haifeng Li, Haiyan Hu, Xifeng Yan, Michael S. Waterman, Haiyan Huang, and Xianghong Jasmine Zhou. Systematic discovery of functional modules and context-specific functional annotation of human genome. *Bioinformatics*, 23(13):i222–i229, 2007.

[19] Earl B. Hunt, J Marin, and Philip J. Stone. *Experiments in Induction*. Academic Press, New York, 1996.

[20] Trey Ideker, Owen Ozier, Benno Schwikowski, and Andrew F. Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, 18(Suppl 1):S233–40, 2002.

[21] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An arpiori-based algorithm for mining frequent substructures from graph data. *Proceedings of 4th European Conference on Principles and Practice of knowledge Discovery in Database (PKDD'00)*, pages 13–23, 2000.

[22] Chuntao Jiang, Frans Coenen, and Michele Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(1):75–105, 2013.

[23] Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Kanae Morishima. Kegg: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Research*, 45(D1):D353–D361, 2017.

[24] Minoru Kanehisa and Susumu Goto. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

[25] Minoru Kanehisa, Yoko Sato, Masayuki Kawashima, Miho Furumichi, and Mao Tanabe. Kegg as a reference resource for gene and protein annotation. *Nucleic Acids Research*, 44(D1):D457–D462, 2016.

[26] Navjot Kaur, Jaspreet Kaur Sahiwal, and Navneet Kaur. Clustering in data mining. *nternational Journal of Computer Trends and Technology (IJCTT)*, 4(5):710–714, 2013.

[27] D Koller, A Battle, and S Mostafavi. Sharing and specificity of co-expression networks across 35 human tissues. *PLoS biology*, 11(5):e1004220, 2015.

[28] Mehmet Koyutürk, Ananth Grama, and Wojciech Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *Bioinformatics*, 20:i200–i207, August 2004.

[29] Mehmet Koyutürk, Yohan Kim, Shankar Subramaniam, Wojciech Szpankowski, and Ananth Grama. Detecting conserved interaction patterns in biological networks. *J Comput Biol.*, 13:1299–322, 2006.

[30] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Subspace clustering. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, 2(4):351–364, July 2012.

[31] M Kuramochi and G Karypis. Frequent subgraph discovery. *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*, pages 313–320, 2001.

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

[33] Wenyuan Li, Haiyan Hu, Yu Huang, Haifeng Li, Michael R. Mehan, Juan Nunez-Iglesias, Min Xu, Xifeng Yan, and Xianghong Jasmine Zhou. *Pattern Mining Across Many Massive Biological Networks*, pages 137–170. Springer New York, New York, NY, 2012.

[34] Siegfried Nijssen and Joost N. Kok. The gaston tool for frequent subgraph mining. *Electron. Notes Theor. Comput. Sci.*, 127(1):77–87, March 2005.

[35] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: A review. *SIGKDD Explor. Newsl.*, 6(1):90–105, June 2004.

[36] Emma Pierson, the GTEx Consortium, Daphne Koller, Alexis Battle, and Sara Mostafavi. Sharing and specificity of co-expression networks across 35 human tissues. *PLOS Computational Biology*, April 2015.

[37] Janet Piñero, Núria Queralt-Rosinach, Àlex Bravo, Jordi Deu-Pons, Anna Bauer-Mehren, Martin Baron, Ferran Sanz, and Laura I. Furlong. Disgenet: a discovery platform for the dynamical exploration of human diseases and their genes. *Database*, 2015:bav028, 2015.

[38] Liva Ralaivola, Sanjay J. Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093 – 1110, 2005.

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

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

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

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

[43] Victor Spirin and Leonid A Mirny. Protein complexes and functional modules in molecular networks. *Proc Natl Acad Sci U S A*, 100(21):12123–12128, 2003.

[44] X Yan and J Han. gspan: Graph-based substructure pattern mining. *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, pages 721–724, 2002.

[45] Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Trans. on Knowl. and Data Eng.*, 12(3):372–390, May 2000.

[46] Mohammed J Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. New algorithms for fast discovery of association rules. Technical report, Rochester, NY, USA, 1997.

[47] Mohamed ZaÃ¯t and Hammou Messatfa. A comparative study of clustering methods. *Future Generation Computer Systems*, 13(2):149 – 159, 1997. Data Mining.

[48] Y. Zhu, X. F. Zhang, D. Q. Dai, and M. Y. Wu. Identifying spurious interactions and predicting missing interactions in the protein-protein interaction networks via a generative network model. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(1):219–225, Jan 2013.