MINING CONNECTED FREQUENT BOOLEAN EXPRESSIONS

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

By

Deepak Kolte

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

Major Department:
Computer Science

May 2017

Fargo, North Dakota

NORTH DAKOTA STATE UNIVERSITY

Graduate School

Title

MINING CONNECTED FREQUENT BOOLEAN EXPRESSIONS

By

Deepak Kolte

The supervisory committee certifies that this paper 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. Oksana Myronovych

Dr. Kambiz Farahmand

Dr. Gursimran Walia

Approved:

21 April 2017
Date

Dr. Brian M. Slator
Department Chair

ABSTRACT

In this paper, we are finding Connected Frequent Boolean Expressions from cancer dataset [14] and protein protein interaction network [14] to discover group of dysregulated genes. Frequent Itemset Mining is a process of finding different sets of items that occur together frequently in a set of transactions. These itemsets are called Frequent Itemsets (FBE).

Connected FBE (CFBE) are a group of items that not only classify as FBE but they are also connected in a graph/network. The nodes in this graph are the items and the edges between them indicate relationships. This can particularly be very helpful in cases where the items are not independent of each other and the presence of one item with another specific item can decide whether the group of items will be frequent or not.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my adviser, Dr. Saeed Salem. Only under his guidance and motivation I was able to complete this research. His knowledge about the subject is phenomenal. I also want to thank Dr. Kendal Nygard for being a great mentor throughout my graduate education. I also thank my committee members, Dr. Gursimran Walia, Dr. Oksana Myronovych and Dr. Kambiz Farahmand, for putting the effort into analyzing my research and complete my graduation. Finally I want to thank North Dakota State University and the Department of Computer Science for the great international exposure and the chance I got to work with some of the brilliant minds known to world.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Finding patterns and association rules is an active research area in many application domains, including bio-informatics, market analysis and cyber security. In early days, this was limited to finding the raw set of items frequent in a given set of transactions in market analysis. These sets of items were called frequent itemset and were of prime interest, since these items can be coupled together better to boost the sales.

Although frequent itemset tells us what items appeared together frequently in a given set of transactions, it doesn't mean they are necessarily associated with each other. Lets say we have a set of transactions from a famous grocery store. When we do the analysis, we found out that shaving gel, shaving blades, peanut butter and bread is a frequent itemset in those transactions. So the owner of grocery store will start placing peanut butter and bread in the beauty section of the store which might not help. The thing we are lacking in this case is the association between those items.

If we had a graph data set consisting of peanut butter, bread, shaving blade and shaving gel as vertices, bread and peanut butter would be connected since they are both food and shaving gel and shaving blades will be connected since they are beauty accessories. But these groups will not be be connected to each other since they are not associated.

Also in case gene networks, the presence of a gene with another specific gene does not lead to a disease but the presence of the same gene with some other gene can contribute to the disease. Such set of genes cannot be determined by mining frequent itemset alone.

The above example shows a frequent itemset might not be used even if it's frequent. In this paper, we first find frequent itemset and then check whether they are connected or not. We call the connected ones as Connected Frequent Boolean Expressions. These $CFBEs$ are not only frequent but are also associated with each other, giving us useful patterns. We can further increase the density of this information by finding boolean expressions which are not only frequent and connected in a given graph but many graphs. And in this information age, where almost everything we touch is designed to emit data about itself, it's very possible to generate graphs for them. We are applying our proposed solution to a gene network. We want to find Frequent Boolean Expressions and then Connected Frequent Boolean Expressions from it. Although the Connected Frequent Boolean Expressions will be useful, we also want to measure how much percent of Frequent Boolean Expressions were actually connected.

We can take example of a group of 20 people who went to a movie festival featuring 20 movies. If we find the group of people who watched at least 12 movies together out of 20, then we can recommend them to each other as friends since they watched at least same 12 movies together. The other way of looking at this would be if most friends out of a group of friends watched at least same 12 movies, then we can suggest this movie to other friends who didn't watched as many as 12 movies.

In Section 2 we summarize the related work already done on this subject. In Section 4 we introduce our proposed algorithm. In Section 6 we apply our algorithm on a gene data set and compare results. In Section 8 we draw conclusions based upon the facts observed in section 6. And finally in section 10 we underline the areas which can be expanded based upon our research.

2. RELATED WORK

2.1. Brute Force Algorithm

Brute Force algorithm [4] is the most basic way of mining Frequent itemset. In this algorithm, we first determine all the candidate itemsets. Then for each itemset, we scan the database of transactions and compute their frequency and then decide if they are frequent or not. Since this method generates all itemset and scans databases iteratively, it is computationally very expensive and redundant as well. Complexity for this method is $O(|I|.|T|.|2^I|)$ and requires $O(|2^I|)$ scans of transaction database D. I is the set of Items, T is the set of transactions. Figure 2.1[4] shows application of the algorithm to dataset in table 2.2.

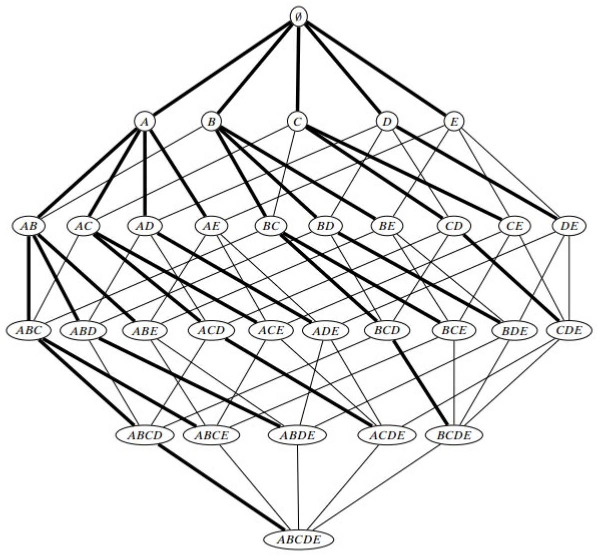


Figure 2.1. Brute force example

2.2. Apriori Algorithm

Apriori algorithm [2, 3, 4] is a traditional algorithm which mines frequent itemset, given a set of transactions and items occurring in them. Apriori algorithm [2, 3, 4] uses downward closure property to remove infrequent itemsets and their super sets from calculations. Another great feature of this algorithm is for generating and counting supports of different itemset, it traverses through them in a Bread-First fashion where the nodes are the itemsets. So a single scan of database computes supports for each of the itemset on a given level in BFS tree rather that scanning the database for each of the itemset on a given level of BFS tree. Each node in the tree represents a itemset and it's frequency in the given set of transactions. Complexity for this method is $O(|I|.|T|.|2^I|)$ but only needs $O(|I|)$ scans of transaction database D. As mentioned before, I is set of Items, T is set of transactions. lin et. al. [13] proposed 3 algorithms similar to Apriori that can work on MapReduce.

This algorithm however is still quite inefficient since it mines all $FBEs$ which are not needed since subsets of a given Frequent Itemset are frequent. Since it calculates all possible candidate itemsets at each stage and is exponential in nature, it becomes very inefficient for even small number of items. In addition since it makes multiple scans of database, the run time and performance goes down for dense data sets where each transaction container larger number of items.

Table 2.1. Sample tabular data set consisting of transactions T1-T12 and items A-I

Transactions/Items	A	B	C	D	E	F	G	H	I	J	K	L	M
T1	1	1	1	1	1	1	1	1	1	1	0	0	0
T2	1	1	1	1	1	1	1	1	1	1	0	0	0
T3	1	1	1	1	1	1	1	1	1	1	0	0	0
T4	1	1	1	1	1	1	1	1	1	1	0	0	0
T5	1	1	1	1	1	1	1	1	1	1	0	0	0
T6	0	0	0	0	0	0	0	0	0	0	1	1	1
T7	0	0	0	0	0	0	0	0	0	0	1	1	1
T8	0	0	0	0	0	0	0	0	0	0	1	1	1
T9	0	0	0	0	0	0	0	0	0	0	1	1	1
T10	0	0	0	0	0	0	0	0	0	0	1	1	1

If we apply Apriori algorithm [2, 3, 4] to the sample data set in table 1 to mine $FBEs$ for support 5, the number would be big as shown. On top of that it will end up scanning the database 10 times in worst case. And this is such a small data set. A real data set would cause serious performance issues with this algorithm and so although this algorithm is better compared to Brute Force, it still is infeasible for real data sets. Moreover we don't need to know all $FBEs$, we just need the maximal ones. For ex. AB, AC, ABC, ABCDEF, ABCEDEFGHIJ are all Frequent Itemsets but we only know that if ABCDEFGHIJ is an FBE then it's subsets would automatically be $FBEs$ as well. Figure 2.2[4] shows application of algorithm to dataset in table 2.2.

No. of Frequent Itemsets $= \sum_{k=1}^{10} {}^{10}C_k + \sum_{k=1}^{3} {}^{3}C_k$

No. of Scan of Database $= |SizeofLargestItemset| = |ABCDEFGHIJ| = 10$

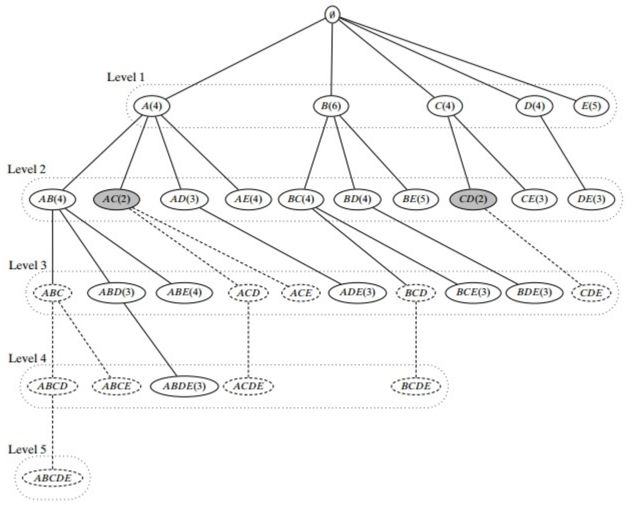


Figure 2.2. Apriori example

2.3. Eclat Algorithm

Eclat algorithm [5] improves upon Apriori [2, 3, 4] by mining the $FBEs$ in a vertical format. It also generates a BFS tree where each node represents an itemset. But instead of storing the frequency, it stores the transactions in which the itemset appears i.e. tid sets. To calculate frequency of any parent itemset, it takes the intersections of the tid sets of child itemset. So it just needs 1 scan of the database. dEclat algorithm [5] is also another a variant of Eclat algorithm [5], where instead of storing list of transactions in each node, we instead store the differential tid sets. In other words, we only store the difference of list of transactions between the child itemset and their parent transaction. The indexing coupled with the differential feature improves storage efficiency further.

If we apply eclat algorithm [5], we will first calculate tids for for all the items and then we will intersect the tids of each items to mine $FBEs$. Figure 2.3[4] shows application of algorithm to dataset in table 2.2.

Table 2.2. Sample tabular data set consisting of transactions T1-T6 and items A-E

Transactions/Items	A	B	C	D	E
T1	1	1	0	1	1
T2	0	1	1	0	1
T3	1	1	0	1	1
T4	1	1	1	0	1
T5	1	1	1	1	1
T6	0	1	1	1	0

Table 2.3. Vertical database generated from Sample tabular data set II

Items	Transactions
A	T1,T3,T4,T5
B	T1,T2,T3,T4,T5,T6
C	T2,T4,T5,T6
D	T1,T3,T5,T6
E	T1,T2,T3,T4,T5

2.4. Frequent Pattern Tree Algorithm

Another algorithm for calculating frequent itemset is Frequent Pattern Tree Algorithm [4]. This algorithm indexes and is more efficient. Each node stores a item instead of storing the whole itemset. Path from root to any node represents the itemset. Also the frequency counts are stored at each node. The frequency count of any given itemset can be found by traversing the path from root to bottom, along the line of items of that itemset and finding the least frequency found at all the individual nodes. Another algorithm [9] uses interim trees to store partial supports. [11] used FP-Trees to optimize candidate set generation for mining frequent itemsets. It doesn't do candidate set generation. It only scans the database twice.

2.5. Maximal Frequent Itemset Mining Algorithms

GenMax algorithm [1] provides great efficiency since it doesn't calculate all $FBEs$ but only the maximal ones. A frequent itemset is called maximal if it doesn't have any other frequent super sets. The base algorithm is same as dEclat [5] but it involves additional checks to prune the itemsets if they already have a frequent superset. Figure [**?**][4] shows the basic algorithm

Figure 2.5[4] shows the application of genmax algorithm [1] to sample tabular data in table 2.2. We get two maximal frequent itemsets - ABDE and BCE. Although we can get all frequent itemsets from the maximal frequent itemsets but not the support of individual frequent itemsets.

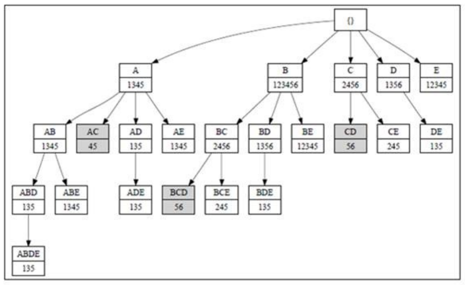


Figure 2.3. Mining FBEs with tid intersection

```
// Initial Call: $\mathcal{M} \leftarrow \emptyset$, $P \leftarrow \{\langle i, \mathbf{t}(i)\rangle \mid i \in \mathcal{I}, sup(i) \geq minsup\}$
GenMax ($P$, $minsup$, $\mathcal{M}$):
$Y \leftarrow \bigcup X_i$
if $\exists Z \in \mathcal{M}$, such that $Y \subseteq Z$ then
    return // prune entire branch
foreach $\langle X_i, \mathbf{t}(X_i)\rangle \in P$ do
    $P_i \leftarrow \emptyset$
    foreach $\langle X_j, \mathbf{t}(X_j)\rangle \in P$, with $j > i$ do
        $X_{ij} \leftarrow X_i \cup X_j$
        $\mathbf{t}(X_{ij}) = \mathbf{t}(X_i) \cap \mathbf{t}(X_j)$
        if $sup(X_{ij}) \geq minsup$ then $P_i \leftarrow P_i \cup \{\langle X_{ij}, \mathbf{t}(X_{ij})\rangle\}$
    if $P_i \neq \emptyset$ then GenMax ($P_i$, $minsup$, $\mathcal{M}$)
    else if $\nexists Z \in \mathcal{M}, X_i \subseteq Z$ then
        $\mathcal{M} = \mathcal{M} \cup X_i$   // add $X_i$ to maximal set
```

Figure 2.4. GenMax algorithm

2.6. Closed Frequent Itemsets Mining Algorithms

CHARM Algorithm differs from GenMax [1] in the fact that it calculates closed frequent itemsets. A frequent itemset is called closed if it doesn't have any frequent supersets with same support. The base algorithm is same as Eclat [5] but it involves additional checks to prune the itemsets if they already have a frequent superset with same support. [12] combined the advantages of CHARM with few other algorithms.

Figure 2.6[4] shows the application of CHARM to sample tabular data set in table 2.2. It must be noted that we can not only generate all the frequent itemsets from closed frequent itemsets

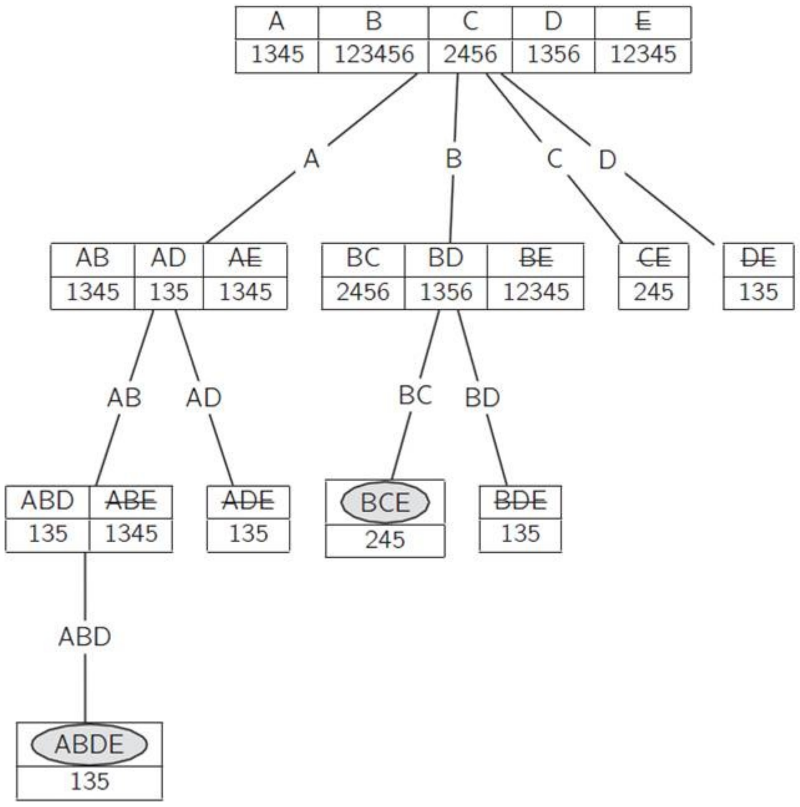


Figure 2.5. GenMax example

but we can also get the individual support numbers for each of the frequent itemsets too. Hence the compression to closed frequent itemsets is a lossless one.

2.7. Weighted Frequent and Closed Itemsets Mining Algorithms

Often times the weights of the itemsets are not equal. In real world, some itemsets can be more important than the others. A weighted algorithm [8] was proposed to calculate weighted frequent closed and maximal patterns. Other area of research is when the trasactions are probabilistic in nature. [10] proposed an algorithm for such kinds of data mining.

2.8. Boolean Expressions

Given a set of transactions and items occurring in them, if we refer the items as literals, then itemset can also be defined as combination of conjunction(OR)and disjunction(AND)of these literals. Boolean Expression are of 4 types, they can be conjunction(OR), disjunction(AND), conjunction of disjunction(OR-AND)and disjunction of conjunction(AND-OR) of literals. Traditional itemsets are $AND - FrequentExpressions$.

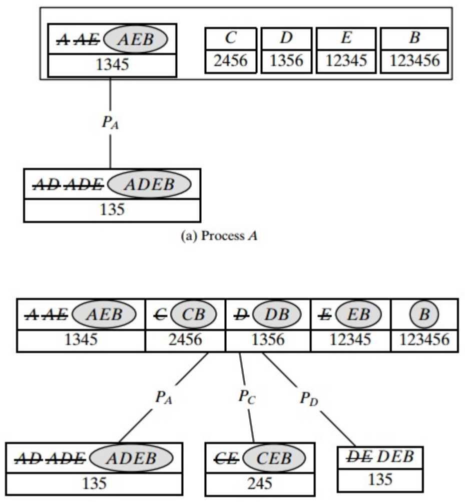


Figure 2.6. CHARM example

2.9. BLOSOM Algorithm

BLOSOM [6] expanded this concept and mines $AND-FBE$, $OR-FBE$, $OR-AND-FBE$ and $AND-OR-FBE$. BLOSOM [6] applies a number of pruning strategies while calculating FBE and so is also efficient.

In this research, we are calculating $CFBE$. This adds another dimension to FBE, since they get filtered out on the basis of a network data. Such behavior is needed where the items in the transactions show very specific behavior with other items. A very good example is gene networks where the presence of a gene in presence of another gene can cause a disease but won't contribute to the disease at all with some other gene.

CoDeNE [7] also mines similar information but with different techniques.

3. PRELIMINARY CONCEPTS

3.1. Itemset

Given a set of transactions and items occurring in them, group of 1 or more items is called itemset. So if there are n items, then theoretically there are be 2^n itemset.

3.2. Frequent Itemset

Given a set of n transactions and k items occurring in them, if we decide m to be minimum support criterion for a frequent itemset ($m <= n$), then if a itemset occurs in at least m transactions, it will be a Frequent itemset.

3.3. Frequent Itemset Mining

Frequent itemset Classification is the process of finding Frequent itemset, given a set of transaction and the items occurring in them. The most famous and simple algorithm to find Frequent itemset is Apriori Algorithm [2, 3, 4]. One of the biggest challenges is how to reduce the search space i.e. number of itemset grow exponentially with number of items. Apriori algorithm [2, 3, 4] uses downward closure property to eliminate candidate itemset. This reduces the search space making it more efficient. Another algorithm is Eclat algorithm [5].

3.4. Boolean Expression (BE)

Given a set of transactions and items occurring in them, if we refer the items as literals, then itemset can also be defined as combination of conjunction(OR)and disjunction(AND)of these literals. Boolean Expression are of 4 types, they can be conjunction(OR), disjunction(AND), conjunction of disjunction(OR-AND)and disjunction of conjunction(AND-OR) of literals.

3.5. Frequent Boolean Expression (FBE)

Given a set of n transactions and k items/literals occurring in them, if we decide the minimum threshold to be m (m¡=n), then if a Boolean Expression occurs in at least n transactions, it will be a Frequent Boolean Expression. As there are 4 types of Boolean Expressions, so are 4 types of Frequent Boolean Expressions too.

3.6. OR Frequent Boolean Expression ($OR - FBE$)

Given a set of n transactions and k items/literals occurring in them, if we decide the minimum threshold to be m ($m <= n$), then if a OR Boolean Expression occurs in at least n

Table 3.1. Sample tabular data set consisting of transactions T1-T5 and items A-D

Transactions/Items	A	B	C	D	E
T1	1	0	1	0	1
T2	1	0	1	0	1
T3	1	1	0	0	0
T4	1	0	1	1	1
T5	1	0	0	1	1

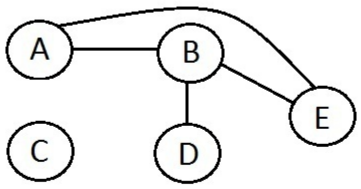


Figure 3.1. Sample graph data set

transactions, it will be a OR Frequent Boolean Expression. The special thing about OR Frequent Boolean Expression is that while counting the frequency of it, we count all those transactions in which at least one of the literals is present. If we mine OR-FBEs for support 50% in above example, then $A|C$ comes out to be an OR-FBE, since it's support is 100%.

3.7. AND Frequent Boolean Expression ($AND - FBE$)

Given a set of n transactions and k *items/literals* occurring in them, if we decide the minimum threshold to be m ($m <= n$), then if a AND Boolean Expression occurs in at least n transactions, it will be a AND Frequent Boolean Expression. AND Frequent Boolean Expression is simply a Frequent itemset since we count only those transactions in which all of the literals of AND Boolean Expression are present. If we mine AND-FBEs for support 50% in above example, then $A\&C$ comes out to be an AND-FBE, since it's support is 60% while B&C is not since it's support is only 20%.

3.8. OR-AND Frequent Boolean Expression ($OR - AND - FBE$)

Given a set of n transactions and k items/literals occurring in them, if we decide the minimum threshold to be m ($m <= n$), then if a OR-AND Boolean Expression occurs in at least n transactions, it will be a OR-AND Frequent Boolean Expression. If we mine OR-AND-FBEs for support 50%, then $(A\&B)|(A\&C)$ is OR-AND-FBE since it's support is 80% while $(A\&B)|(B\&C)$ is not, since it's support is only 20%.

3.9. AND-OR Frequent Boolean Expression ($AND - OR - FBE$)

Given a set of n transactions and k items/literals occurring in them, if we decide the minimum threshold to be m ($m <= n$), then if a AND-OR Boolean Expression occurs in at least n transactions, it will be a AND-OR Frequent Boolean Expression. . If we mine AND-OR-FBEs for support 50%,then $(A|B)\&(A|C)$ is AND-OR-FBE since it's support is 80% while $(B|C)\&(B|D)$ is not, since it's support is only 40%.

3.10. Downward Closure Property of Frequent itemset

Downward closure property implies that if a given itemset consisting of n items is frequent, then all the possible itemsets of n-1 items that make up the n itemset are frequent too. In other words if one or more than one of n-1 itemset are not frequent, then the itemset of n items formed from these items will not be a frequent itemset too. For ex. If ABCE is an frequent itemset, then ABE, ACE, BCE must be frequent itemset too. On the other hand if one or more of ABE, ACE and BCE are not frequent itemset then ABCE will not be a frequent itemset too.

3.11. Maximal Frequent Boolean Expression

A frequent itemset is called maximal frequent itemset if it doesn't have any frequent supersets. Lets say T is the transaction set, I be the itemset, M is the set of maximal frequent boolean expressions and F be the set of frequent boolean expressions, then

F = $\{X|X \subset I \ and \ support(X) \geq s_{min}\}$

M = $\{X \in X \subset F \ and \ \nexists Y \supset X, such \ that \ Y \in F\}$

3.12. Closed Frequent Boolean Expression

A frequent itemset is called closed frequent itemset if it doesn't have any frequent supersets with same support. Lets say T is the transaction set, I be the itemset, C is the set of closed frequent boolean expressions and F be the set of frequent boolean expressions, then

F = $\{X|X \subset I \ and \ support(X) \geq s_{min}\}$

C = $\{X|X \in F \ and \ \nexists Y \supset X, such \ that \ support(X) = support(Y)\}$

4. PROPOSED ALGORITHM

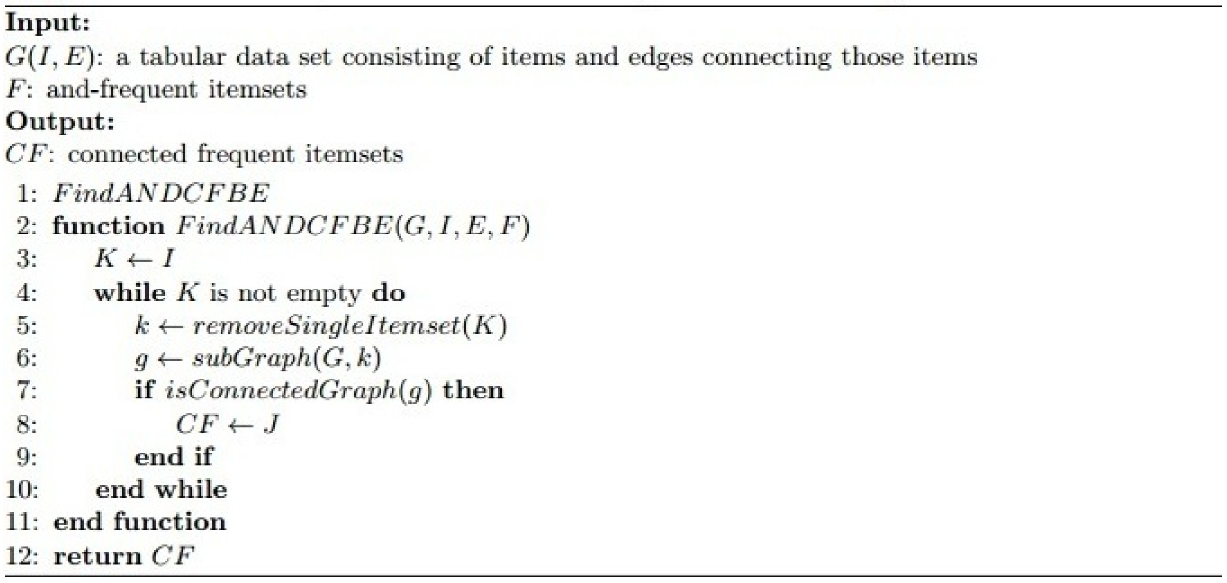
In this section we introduce our algorithm. It is a two phase algorithm.

4.1. Finding AND-Frequent Boolean Expression

In this section, we have a gene data set, which consists of genes observed in a given number of patients. It's a tabular data and gene is marked present in a cell if it contributed to the disease for that patient and is marked absent from the cell if it did not contribute to the disease. We use genmax algorithm [1] to mine $AND - FBE$ from this data set and hence we only get maximal frequent boolean expressions. We have already explained GenMax algorithm [1] in detail before.

4.2. Finding AND-Connected Frequent Boolean Expression

In this section, we take each of the $AND - FBEs$ mined in section 4.1 and then mine $AND - CFBEs$ from them. Here we have a graph data set G, where genes are the vertices/items and edges connecting these vertices signify the relationship between them. For each $AND - FBE$, we grab each of the items that the given $AND - FBE$ is made up, induce a sub-graph $g(G, e)$ containing only these vertices and then check whether the formed sub graph $g(G, e)$ is connected or not. The sub graph $g(G, e)$ which is connected, we label the corresponding $AND - FBE$ as $AND - CFBE$. The basic algorithm is shown below



Input:
$G(I, E)$: a tabular data set consisting of items and edges connecting those items
F: and-frequent itemsets
Output:
CF: connected frequent itemsets

```
$FindANDCFBE$
function $FindANDCFBE(G, I, E, F)$
    $K \leftarrow I$
    while $K$ is not empty do
        $k \leftarrow removeSingleItemset(K)$
        $g \leftarrow subGraph(G, k)$
        if $isConnectedGraph(g)$ then
            $CF \leftarrow J$
        end if
    end while
end function
return $CF$
```

Figure 4.1. Algorithm 1

5. PROPOSED ALGORITHM FOR FINDING CFBE FOR SAMPLE DATA SET

In this section, we pick up each $AND - FBE$ obtained before and induce a subgraph for the items in that $AND - FBE$. If the induced sub graph is connected, then we that $AND - FBE$ qualifies as a $AND - CFBE$, else we reject it. We use dataset in table 4.1 and graph in figure 4.1.

Table 5.1. $AND - FBE$

$AND - FBE$	ABDE,BCE

For an example, we will check whether or not $ABDE$ is a $AND - CFBE$. First we find the subgraph for items A,B,D,E. Since that graph is connected, that means $ABDE$ is a $AND - CFBE$.

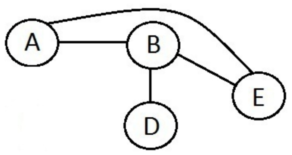


Figure 5.1. SubGraph ACE

Another example, we can check whether or not BCE is a $AND - CFBE$. First we find the subgraph for items B,C,E. Since that graph is not connected, that means BCE is not a $AND - CFBE$.

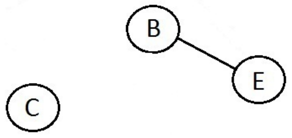


Figure 5.2. SubGraph ADE

6. EXPERIMENTS

We used cancer data set which had 23702 genes that were observed in 13 subjects [14]. So 23702 items were observed in 13 transactions. This data set was used to mine $AND - FBEs$ in phase 1 of the algorithm. The other data set we used was from same subjects, this data set is actually a graph where nodes of the graph are the genes and the links connecting them represents whether or not they are associated. This graph data set along with the $AND - FBEs$ mined was then used to mine $AND - CFBEs$ in phase 3 of the algorithm.

The limitation in the whole algorithm is phase 1, where the candidate Boolean Expressions are so many, that the algorithm just cannot find all of AND-FBEs and gets stuck. At that point, we have to pick up whatever $AND - FBEs$ the phase 1 mined and feed into phase 2 of the algorithm.

Experiments were conducted on an windows machine with Intel Core i7-3630QM (2.4GHz) processor and 8 Gigabytes of main memory.

7. RESULTS

7.1. FBEs

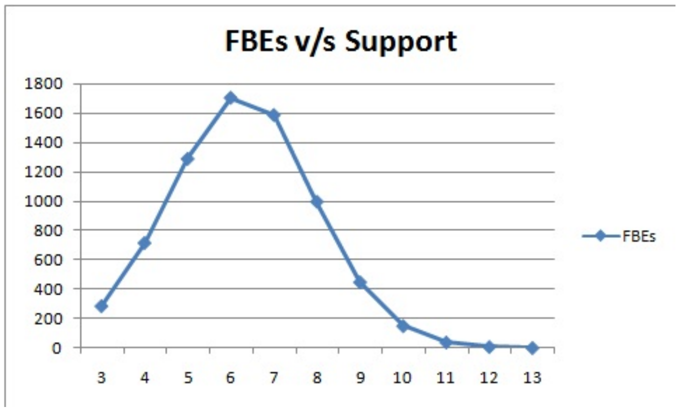


Figure 7.1. AND-FBEs for different support

From this graph, we can see that as we start from minimum support 3, the numbers of $FBEs$ found are low. This is due to the fact that our algorithm returns only maximal $FBEs$, so at lower support, most of $FBEs$ gets consolidated in a higher $FBEs$ since everything gets counted as frequent. Fro ex. if AB, BC, $ABCDE$ are all frequent, AB and BC won't show up since they are part of $ABCDE$. So the $FBEs$ found at lower supports contain more literals and are fewer in number. As we increase minimum support, higher $FBEs$ won't be counted as frequent anymore and so their children which are frequent, get counted and $FBEs$ found are higher in number, though they contain less literals. For ex. AB, BC now will now show up as $FBEs$ since ABC is not frequent anymore but AB and BC are. This trend continues until the number of $FBEs$ found reaches a peak of 1702 at support 6. After this another factor comes into play. As we are increasing the minimum support requirement, even the smaller $FBEs$ are not being counted as frequent. For ex AB might be frequent but not BC, so the $FBEs$ found now are lesser in number and this trend continues for the rest of the experiment which ends at support 13.

7.2. AND-CFBE

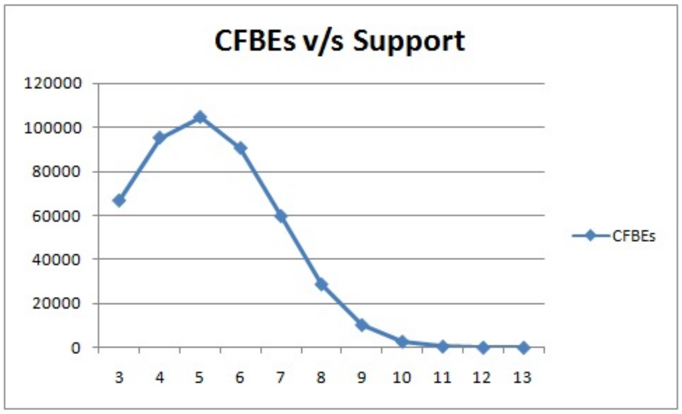


Figure 7.2. AND-CFBEs for different support

From this graph, we can see that as we start from minimum support 3, the numbers of $CFBEs$ found are low. This is similar trend and also the same reason that we observed while finding out $FBEs$. Since the algorithm returns only maximal $CFBEs$, so at lower support, most of $CFBEs$ gets consolidated in a higher $CFBEs$ since everything gets counted as connected. Fro ex. if AB, BC, $ABCDE$ are $FBEs$ and they all connected too then, AB and BC won't show up as since they will be part of $ABCDE$. So the $CFBEs$ found at lower supports contain more literals and are fewer in number. As we increase minimum support, higher $CFBEs$ won't be counted as connected anymore and so their children which are connected, get counted and $CFBEs$ found are higher in number, though they contain less literals. For ex. AB, BC will now show up as $CFBEs$ since ABC is not connected anymore but AB and BC are. This trend continues until the number of $CFBEs$ found reaches a peak of 104612 at support 5. After this another factor comes into play. As we are increasing the minimum support requirement, even the smaller $CFBEs$ are not being counted as connected. For ex AB, BC are frequent and AB is connected but not AB. So the $CFBEs$ found now are lesser in number and this trend continues for the rest of the experiment which ends at support 13.

The other thing to note is that CFBEs are always a small percent of the found FBEs.

All of the above graphs precisely bring us to the point we were making earlier. Just because some literals appear in set of transactions doesn't necessarily mean they are associated. And this information becomes very important where the presence of an item can completely change the behavior of itself in the presence or absence of another item in same set of transactions. Traditional $AND-FBEs$ only focus on the raw frequency to mine $AND-FBEs$ but we go one step further to check whether or not these $AND-FBEs$ are connected/associated or not. The associated ones are called $AND-CFBEs$ and they are not only frequent but they are also associated to each other making it easier to draw conclusions about a group of genes triggering a particular behavior.

Table 7.1. Statistics of CFBEs found

Support	No. of FBEs	No. of CFBEs per No. of FBEs	Avg. Size of CFBEs
3	286	23.33	1.53
4	715	133.20	1.40
5	1287	81.28	1.33
6	1702	53.21	1.28
7	1585	37.66	1.25
8	996	28.90	1.23
9	448	23.07	1.21
10	150	18.60	1.17
11	39	14.69	1.12
12	8	10.65	1.06
13	1	7.00	1.00

The only clear trend that we can see from the statistics is that as we go for higher and higher support, the sizes of $CFBEs$ decreases.

8. CONCLUSION

The most general conclusion that can be drawn is only a little fraction of $AND-FBE$ are $AND-CFBE$.

Secondly, as we increase the minimum support threshold, the number of AND-FBE decreases and the number of AND-CFBE decreases too.

Thirdly, since there are only 13 transactions, the graph is not smooth and it has discrete breakpoints. For ex. the number of AND-FBE with 100% support will be same as the number of FBE with 99% support, since [13*100/100] and [13*99/100] are both 12 and so is [13*93/100]. But the breakpoint happens at 92% when [13*92/100] = 11. Similarly, the other breakpoints are at 7%, 15%, 23%, 30%, 38%, 46%, 53%, 61%, 69%, 76%, 84%, 92% & 100%.

Fourth, as we search for bigger and bigger $AND-FBEs$ (which contain more literals), the number of such $AND-FBEs$ are less in number. For ex. The number of $3-AND-FBEs$ will be more or equal to number of $4-AND-FBEs$ and the number of $4-AND-FBEs$ will be more than or equal to the number of $5-AND-FBEs$.

9. APPLICATIONS

First and foremost usage is in market basket analysis to identify consumer trends. This is what frequent pattern mining originally was created for. Giant stores and corporations has tons of data regarding customer spending. They are always keen in knowing what what sells more and what sells less , so they can package and bundle their products well.

It is also used for mining data on web, since web is full of different sorts of graphs. A specific case would be as a recommended system for social networks. If we consider users as items and they can have different attributes like some like comedy movies, some like action movies and others like fantasy. Then if there is a movie event or festival around and these users participate in that event, so those will be trasactions. We can then mine frequent itemsets, which will be a maximal or closed set of users. Now if we pick up one of this frequent itemsets of users, not all the users will have same attributes, which in this case is the kind of movies they like. The minority user in a itemset who do not share the same attribute can be recommended those attributes.

Software bug detection is another interesting application of this algorithm. Software consists of different module, which can be considered items and bugs can be considered as transactions. More over there are functional dependencies between different module which can be described as a graph. So we can mine frequent itemsets, it will actually give us frequent sets of modules involved when the bug was caused. In addition we can make this information more reliable, when we only consider the module which are connected in the dependency for a given frequent itemset.

Cyber security is another active area of research for frequent itemsets. One of the prominent attacks are Distributed Denial Of Service (DDOS). It involves group on compromised computers to spam request messages to a target machine, rendering it useless for anyone else. If we consider these compromised machines as items and these attacks as transactions, then we can mine frequent itemsets or the machines that are compromised frequently and cause such damage. They can either be the source or can help track the source since they are involved frequently in such attack.

10. FUTURE WORK

There are some major areas where our research can be expanded. First, we can extend the logic of phase II of our algorithm to mine all types of boolean expressions i.e. $AND-CFBE$,$OR-CFBE$,$AND-OR-CFBE$,$OR-AND-CFBE$ and not just limited to $AND-CFBE$.

Second, we can make further changes so that the algorithm can work with negative literals as well.

Third, we can get more meaningful expressions if we use multiple set of graphs to check the connectivity of boolean expressions.

Fourth, we can take this algorithm one step further to make it a recommendation system in social networks. We can also apply the idea in a reverse manner to find out whether people sharing common interests or activities are connected or not.

Also, this research can be applied to a graph dataset containing more than one attribute. It can also be applied to multiple graph of single attribute and then finding common sub-graphs in each of them.

REFERENCES

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

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB ’94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[3] Hannu Toivonen. *Apriori Algorithm*, pages 39–40. Springer US, Boston, MA, 2010.

[4] Mohammed J. Zaki and Jr. Wagner Meira. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, May 2014.

[5] Mohammed J. Zaki and Karam Gouda. Fast vertical mining using diffsets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’03, pages 326–335, New York, NY, USA, 2003. ACM.

[6] Lizhuang Zhao, Mohammed J. Zaki, and Naren Ramakrishnan. Blosom: A framework for mining arbitrary boolean expressions over attribute sets. In *Proceedings of of the 12th International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 827–832, 2006.

[7] Brazier T. Goparaju A. and Salem S. Mining reprensentative cohesive dense subgraphs. *Network Modeling Analysis in Health Informatics and Bioinformatics, 4(1), pp. 1-11, 2015.*, 2015.

[8] Unil Yun and Keun Ho Ryu. Approximate weighted frequent pattern mining with/without noisy environments. *Know.-Based Syst.*, 24(1):73–82, February 2011.

[9] F. Coenen, P. Leng, A. Pagourtzis, W. Rytter, and D. Souliou. Improved methods for extracting frequent itemsets from interim-support trees. *Software: Practice and Experience*, 39(6):551–571, 2009.

[10] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, and Andreas Zuefle. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pages 119–128, New York, NY, USA, 2009. ACM.

[11] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[12] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 236–245, New York, NY, USA, 2003. ACM.

[13] Ming-Yen Lin, Pei-Yu Lee, and Sue-Chen Hsueh. Apriori-based frequent itemset mining algorithms on mapreduce. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '12, pages 76:1–76:8, New York, NY, USA, 2012. ACM.

[14] Stark C, Breitkreutz B-J, Reguly T, Boucher L, Breitkreutz, and Tyers M. Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*, pages 34(Database issue):D535–D539, 2006.