Doctoral Dissertation

Development of an Integrated System for Clustering of Simple and Bipartite Graphs And its Application to Different Types of Biological Data

Mohammad Bozlul Karim

Sep 11, 2019

Department of Information Science
Graduate School of Information Science
Nara Institute of Science and Technology
Japan

A Doctoral Dissertation

submitted to the Graduate School of Information Science,

Nara Institute of Science and Technology

in partial fulfillment of the requirements for the degree of

Doctor of Engineering

Thesis Committee:

Professor Shigehiko Kanaya	(Supervisor)
Professor Kenichi Matsumoto	(Co-supervisor)
Associate Professor Md. Altaf-Ul-Amin	(Co-supervisor)
Associate Professor Naoaki Ono	(Co-supervisor)

Development of an Integrated System for Clustering of Simple and Bipartite Graphs And its Application to Different Types of Biological Data

Mohammad Bozlul Karim

Abstract

Network analysis particularly graph clustering has become a useful and important technique in data mining applications. It provides a global view of data structure where highly concentrated data are grouped based on their common properties. We propose a novel biclustering approach called BiClusO. We compare our biclustering algorithm with five different algorithms using biological and synthetic data and evaluate the performances. Our algorithm shows the best performance over the selected five biclustering algorithms. We also present new integrated software implementing the DPClusO and BiClusO algorithms to be utilized for simple and bipartite graph clustering. This tool provides the user with GUI based facilities for simple and bipartite graph clustering along with filtering and amalgamation of clusters, hierarchical node analysis, node distribution among cluster set and visualization of all or partial portion of a big cluster set. We used this tool to analyze the bipartite relations between species and volatile organic compounds (VOCs). VOCs emitted by different species have huge environmental and ecological impacts. Biosynthesis of VOCs depends on different metabolic pathways based on which the species can be categorized. Our experiment shows that VOC based classification is consistent with taxonomy based classification of the species. We assessed the diversity of VOC pathways across different species classes by using structurally similar VOC groups (SSVGs). We also analyzed the mRNA and miRNA bipartite relations. In such relation a miRNA-regulatory module (MRM) is a subset of MTIs where a groups of miRNAs participate cooperatively by regulating a bunch of genes to control different biological processes. We mainly focused on MRMs detection from MTIs involving the IBD related genes. We evaluated the relevance of the miRNAs with IBD by counting their occurrences in different MRMs and their interactions with known IBD genes. Finally we successfully identified some important IBD related miRNAs

Keywords: Cluster, bicluster, volatile organic compound, miRNA, IBD

[1]Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, Sep 11, 2019.

Acknowledgements

First I would like to thank my supervisor, Professor Shigehiko Kanaya for giving the opportunity to complete this research work. His continuous support and motivation help me to achieve this immense goal in my study life.

I want to express my gratitude for Professor Kenichi Matsumoto for giving his time to review my thesis with insightful recommendations.

My sincere thanks go to co-supervisor Associate Professor Md. Altaf-Ul-Amin who inspired me throughout my research time with a lot of help and support. During the predicament period of my research work, he always supports me to be patient and strong.

My special thanks go to Associate Professor Naoaki Ono and Assistant Professor Ming Huang for all their support, valuable comments and suggestions. I am also grateful to Mrs. Minako Ohashi for her help in the administrative matter and Mrs. Aki Hirai Morita for her support for using the KNApSAcK Family Databases.

I wish to thank all my fellow lab members in Computational Systems Biology Laboratory for their technical support. Together we had a lot of fun and celebrated different occasion. I surely will miss the time of staying in this lab.

I would express my gratitude to Prof Dr. Liakot Ali from IICT department of Bangladesh University of Engineering and Technology. His inspiration also helped me to achieve this research work.

I would also like to express my gratitude to the authority of the international division of the Nara Institute of Science and Technology for providing me time to time help and support. With their assistance, I feel like spending my entire research time in a friendly environment. I met a lot of different cultural student and people in here which also helped me to learn a lot of knowledge.

My special and highest appreciation to all my family members, especially my parents, my wife Shamima Momen and my daughters Sujana and Sanjana for their immense support and unconditional love. I also apologize to them not giving enough time due to my research work.

I also express my gratitude to my elder brothers Dr. Kamrul Zaman and Dr. Rezaul Karim for their prayer and encouragement.

Sep 11, 2019

Mohammad Bozlul Karim

List of Abbreviations

VOC	Volatile organic compound
SSVG	Structurally Similar VOC group
mRNA	Messenger RNA
miRNA	Micro RNA
SDF	Structure definition file
APFP	Atom Pair Finger Print
IBD	Inflammatory Bowel disease
MTI	MicroRNA Target Interaction
MRM	MicroRNA Regulatory Module
CD	Crohn's Disease
UC	Ulcerative colitis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Network analysis is a comprehensive term used in different fields of studies such as biology, economics, sociology, geography, social psychology, business study, etc. Nowadays almost every research where a significant amount of relational data are needed to analyze uses network analysis to find out the kernel information. Conventional database structure like a primary-foreign key relationship or even self-reference relationship is not adequate to represent this heterogeneous relationship of a network. Graphical representation of network data using edge set, vertex set is a very useful way to analyze and understand this relationship. Scientists are trying to find many important motifs from the network by applying different simple clustering and biclustering methods. This research focuses on developing a new biclustering algorithm which can generate a substantial number of biclusters from a network by controlling different overlapping parameters. We evaluate the performance of our algorithm by comparing it with five different well-known biclustering algorithms. We apply our biclustering algorithm to analyze species-VOC and miRNA-mRNA bipartite relations. Finally, we develop an integrated system of clustering simple and bipartite graph with different GUI based interactions.

1.1 Application of clustering

Graph clustering has been applied in various fields of research. Graph cluster analysis finds similar group of nodes where each group represents some global measure over a set of characteristics. A simple graph is constructed using a single node set connected by common property. As an example let a graph is constructed where the nodes are people and each of them are connected by a common property i.e. similar height. Clustering such graph may produce different groups of people with similar height. Similarly finding different groups of structurally similar chemical compounds from a graph where chemical compounds are nodes and an edge represents the structural similarity between two compounds.

In a bipartite graph, two features are considered to have a common property. In this type of relation, two different features have many to many relations in terms of common property. There are numerous applications of biclustering in different field of study. In biology, gene expression under certain conditions forms a bipartite network which helps to identify the cellular response, disease diagnosis and pathway analysis [1][2]. Biological network analysis of the pairwise selection of protein, miRNA, metabolite, conserved functional subsequences, and factor binding sites can predict or understand the different cellular mechanism [3]-[5]. In text mining content related document searching is done using biclustering where rows represent different document types and columns represent frequently used keywords in the documents[6][7]. In a social or business network finding a common group of people with different shared interest are analyzed by biclustering[8][9].

1.2 Research objective

Understanding a cluster set from any simple network or bipartite network by using the textual representation or limited graphical options do not provide adequate information about the observed dataset. Even some times a hierarchical association among clusters are necessary to analyze which provides the common inherited properties. In order to grasp the relationships among clusters, overlapping property is an important parameter which needs to be studied. Most of the clustering algorithms have limited options on overlapping controlling parameter. The algorithms proposed in different works of literature have an only textual presentation or limited graphical options. Some third-party tools have been implemented using different clustering or biclustering algorithm emphasizing on the simple implementation of those algorithms. DPClus and DPClusO developed in our lab previously, have been used in different biological data analysis. We develop

BiClusO algorithm and create an integrated system combining those algorithms with filtering and amalgamation of clusters, hierarchical node analysis, node distribution among cluster set and visualization of all or partial portion of a big cluster set. We evaluate the performance of our algorithm with two different biological datasets and four different synthetic datasets. The performance proves the goodness of our algorithm which will attract researchers on biclustering issue.

As an example, we apply our tools to species-VOC bipartite data and find some interesting property which will help to understand the classification of species in terms of VOC and biodiversity of VOC pathway. We also apply our tool to analyze the miRNA-mRNA bipartite data and identify some MRM of Inflammatory bowel disease (IBD). We identify an important set of miRNA from those MRM which have a significant impact on IBD and related disease. Most of these analyses, we use our algorithms and GUI based system to analyze and presentation of the results.

1.3 Outlines

Chapter 2 describes the method of BiClusO. We use two illustrative examples for creating bicluster from the bipartite graph. The pseudocode of the algorithm is described in two different modules. We also explain two different overlapping controlling parameters which are used to control overlapping among simple clusters or biclusters.

Chapter 3 describes the performance evaluation of our algorithm using two different biological dataset and four synthetics dataset. We create four different synthetic datasets using bicluster size and overlapping property. We also use the bipartite relationship of the species-VOC and gene expression data of the C. Cerevisiae. We apply the hypergeometric test and GO enrichment analysis to measure the performance of different algorithms on biological dataset. We use two matrices i.e average cluster relevance and average module recovery to accesses the performance of different algorithms.

In chapter 4, we apply the BiClusO and DPClusO on species-VOC data analysis. We successfully identify some taxonomical species in terms of VOC data. We explain the process of optimal parameter setting for BiClusO. Using DPClusO. We find the structurally similar VOC group (SSVG) and mapped those SSVGs to the different kingdom of species which reveal the diversity of VOC pathway and conservation of evolutionary hierarchy.

In chapter 5, we apply the BiClusO, DPClusO on finding some effective genes and miRNAs related to IBD by analyzing the simple gene regulatory network and mRNA-miRNA bipartite network. First we collect 4477 DEG genes NCBI's Gene Expression Omnibus for both UC and CD and use these genes as seed point to the dataset from HIPPE network. We use DPClusO algorithm and compare the result to four different databases. of HuGENet, DisGeNet, CTD and another genome-wide association study (GWAS). We identify 2245 IBD related genes using DPClusO and those databases. We use those genes as the primary target of miRNA and apply BiClusO to four different datasets of the bipartite relationship of miRNA-mRNA from mirWalk, DIANA, miRecords, and miRTarbase.

Chapter 6 describe the implementation detail of DPClusO and BiClusO using GUI. We explain user interaction with different parameter selection using two illustrative examples. The system provides simple clustering, biclustering, simple cluster hierarchical relation, bicluster hierarchical relation, cluster joining, cluster filtering, visualization of the cluster set using different layout. Partial visualization of a single cluster etc.

Chapter 7 describe the future plan and conclusion

Chapter 2

Development of a new biclustering algorithm

Biclustering is an important data mining technique usually used to partition a sparse data matrix into a finite number of highly dense submatrices. In computational systems biology, two variable pairs such as gene-condition, gene-patient, or gene-disease are widely used with biclustering to identify the specific set of genes up-regulated and/or down-regulated by similar types of conditions, patients, or diseases. Also, some studies use biclustering to classify herbivorous species based on the plants they feed on to understand the evolutionary changes of the species in adapting to the availability of certain kinds of plants in their habitat [10]. Biological network analysis of the pairwise selection of protein, miRNA, metabolite, conserved functional subsequences, and factor binding sites can predict or understand the different cellular mechanism. In text mining content related document searching is done using biclustering where rows represent different document types and columns represent frequently used keywords in the documents. In a social network, finding a common group of people with different shared interest are analyzed by biclustering. Thus biclustering can be applied to various types of bipartite data in different fields.

2.1 Introduction

In this paper, we introduce a new biclustering algorithm called BiClusO using the concept of the Tanimoto coefficient, relation number, and the DPClusO algorithm [11]- [13] and emphasize the construction of a simple graph by means of the first node set of a bipartite graph. In our approach, the edges of the simple graph are selected based on the common neighbors (in the second set) of the nodes of the first set. Thus, we create data folding and apply the DPClusO algorithm to generate a cluster set. After generating a cluster set, we unfold the data by assigning the

members of the second node set to an individual cluster using the probability of their attachment to cluster nodes. Our algorithm can produce overlapping biclusters. Overlapping biclustering means a node of any side of a bipartite graph may belong to more than one bicluster. The problem of finding a minimum bicluster set which is either mutually exclusive or overlapping and covers all data elements from a bipartite graph has been proven to be NP-hard [14]. So, we select a single node set, i.e, the most significant node set, rather than both node sets to convert the biclustering problem to a simple graph clustering problem, for which there are polynomial-time heuristic
algorithms.

2.2 The concept of biclustering

A bipartite graph is a graph that consists of two disjoint sets of nodes say U and V such that each edge connects a node in U with a node in V i.e. U and V are independent sets. Mathematically, a bipartite graph is denoted as $G = (U, V, E)$ where U and V are two disjoint partition in G. An edge matrix is represented by a weight function $w: U \times V \rightarrow \{0,1\}$ such that $w((u,v)) = 1$ for $(u,v) \in E$ and $w((u,v)) = 0$ for $(u,v) \notin E$.

A bicluster represents a pair of subset $U' \subseteq U$ and $V' \subseteq V$ with $w: U' \times V' \rightarrow \{0,1\}$ with higher density of *'1'*. A biclustering algorithm finds out the region bounded by $I = |U'|$ rows and $J = |V'|$ columns which maximize the density and dimension of submatrix. In simple terms, a bipartite graph can be represented by a binary matrix. The concept of biclustering can also be extended for a general matrix. There are several different approaches invented to seek maximal region biclusters [13][15][16][17]. Cheng and Church (2000) applied the greedy approach based biclustering method in gene expression data. Amos, Roded and Shamir developed SAMBA (Statistical-Algorithmic Method for Bicluster Analysis) using the statistical data modeling to calculate vertex pair weighting, hashing technique to find the heaviest bicliques and local score improvement procedure (addition or deletion of a single vertex). FABIA (Factor Analysis for Bicluster Acquisition) uses linear dependencies between samples and feature patterns. It is based on a multiplicative model which captures realistic non-Gaussian data distributions with heavy tails as observed in bipartite relationship of

gene expression data. BiMax uses divide and conquer method recursively to enumerate all the maximal biclusters in a binary data matrix. Another statistically significant biclustering for binary data matrix is defiened by Koyutürk [18] which is reffered to as Cmnk [19]. Most of these approaches do not support clustering by controlling overlapping. Cmnk and BiMax which are mainly designed for binary data set cannot produce good result when there are large number of zeros [19] in the input binary matrix. In this work, we propose a heuristic algorithm for finding biclusters from binary matrix data. Our approach can generate overlapping biclusters by utilizing the overlapping property of DPClusO.

2.3 The biclustering algorithm BiClusO

Table 2.1 summarizes the notations and their meanings that are used in the description of the algorithm.

Table 2. 1 Notation and their meaning.

Notation	Description
U, V	Two disjoint sets of nodes of a bipartite graph
E	Edge set between U and V
$N(u)$	Neighbor of u where $u \in U$ and $N(u) \subseteq V$
$N(v_i k)$	Neighbor of v_i in k^{th} cluster $v_i \in V$, $N(v_i) \subseteq U$ and $\lvert N(v_i k) \rvert \leq \lvert N(v_i) \rvert$
R	Relation matrix with dimension $\lvert U\rvert \times \lvert U\rvert$ where $R_{i,j} = N(u_i) \cap N(u_j)$
T	Tanimoto cofficient matrix with dimension $\lvert U\rvert \times \lvert U\rvert$ where $T_{i,j} = (\lvert R_{i,j}\rvert / \lvert N(u_i) \cup N(u_j)\rvert)$
Th_{rl} $(\in R)$	Relation number threshold, usually any small element from R

	matrix
Th_{tf} $(\in T)$	Tanimoto cofficient threshold, usually any small element from T matrix
F	Boolean matrix of dimension $\|U\| \times \|U\|$ constructed by using $R, T, Th_{rl},$ and Th_{tf} where $F_{i,j} \in \{0,1\}$
Pvk	Probability set for finding second set of nodes in k^{th} cluster. Where $vk=\{v_1, v_2, \dots\dots v_m\} \subseteq V$, the total m number of attached nodes in k^{th} cluster
Pv_ik	Probability of node v_i to be included in k^{th} cluster $Pv_ik \in Pvk$ $1 \leq i \leq \|Pvk\|$
$Pvth$	Threshold value of finding second set of nodes in a cluster, expressed as probability.
VOC_{Ck}	VOC set of k^{th} cluster.

In this work, we use data folding mechanism to create a simple graph G_s from the bipartite graph $G = (U, V, E)$ involving the nodes of set U. We discuss the proposed biclustering algorithm by means of two separate algorithms. The Algorithm 1 creates the relation matrix and the Tanimoto cofficient matrix. Based on these two matrices the second algorithm construct G_s, apply DPClusO to generate clusters and then unfold the data by assigning the members of set V to individual clusters and thus complete the biclusters. The R and T matrices are diagonally symmetrical square matrices and we only need to calculate the lower triangular part of them. Condition $(i<j)$ in step 1 of Algorithm 1 only consider the lower triangular portion of R and T. If we have a total number of d elements in set V then the degree of the nodes in set U may vary between 1 to d considering the fact that there exist no isolated node in U and V. Such degree distribution makes the Algorithm 1 to output relation matrix with the values ranging between $(0, d)$. Here 0 means no common neighbor between two nodes. Step 1 to Step 4 loop through the nodes of set U.

Calculating Step 3 and 4 needs only to find out the union and intersect operation on neighbors of nodes of set U from set V.

2.3.1 Pseudocode of the algorithm

Algorithm 1: Constructing R and T matrices

Input: Bipartite graph $G = (U, V, E)$ represented by a matrix where the rows and columns correspond to the nodes of set U and V respectively.

Output: Two square matrices R and T containing relation number and Tanimoto coefficient between each pair of nodes of U.

Pseudo Code:

```
1. Find node pairs $(u_i\ u_j) \in U$ where $(i < j)$
2. Find neighbors set of $u_i$ and $u_j$ from $V$ as $N(u_i), N(u_j)$
3. Calculate the total number of the common neighbor $|N(u_i) \cap N(u_j)|$
   from $N(u_i)$ , $N(u_j)$ and assign it to relation matrix as $R_{i,j}$
4. Calculate the Tanimoto coefficient $(|N(u_i) \cap N(u_j)| / |N(u_i) \cup N(u_j)|)$
   between $u_i$ and $u_j$ and assign it to Tanimoto matrix as $T_{i,j}$
5. Output the $R$, $T$ matrix
```

Algorithm 2: Generate biclusters

Input: R and T matrices from Algorithm 1, Relation number threshold Th_{rl}, Tanimoto coefficient thereshold Th_{tf}, Cluster Property CP, Cluster Density CD, Overlapping Coefficient OV, $Pvth$

Output: Bicluster set

Pseudo Code:

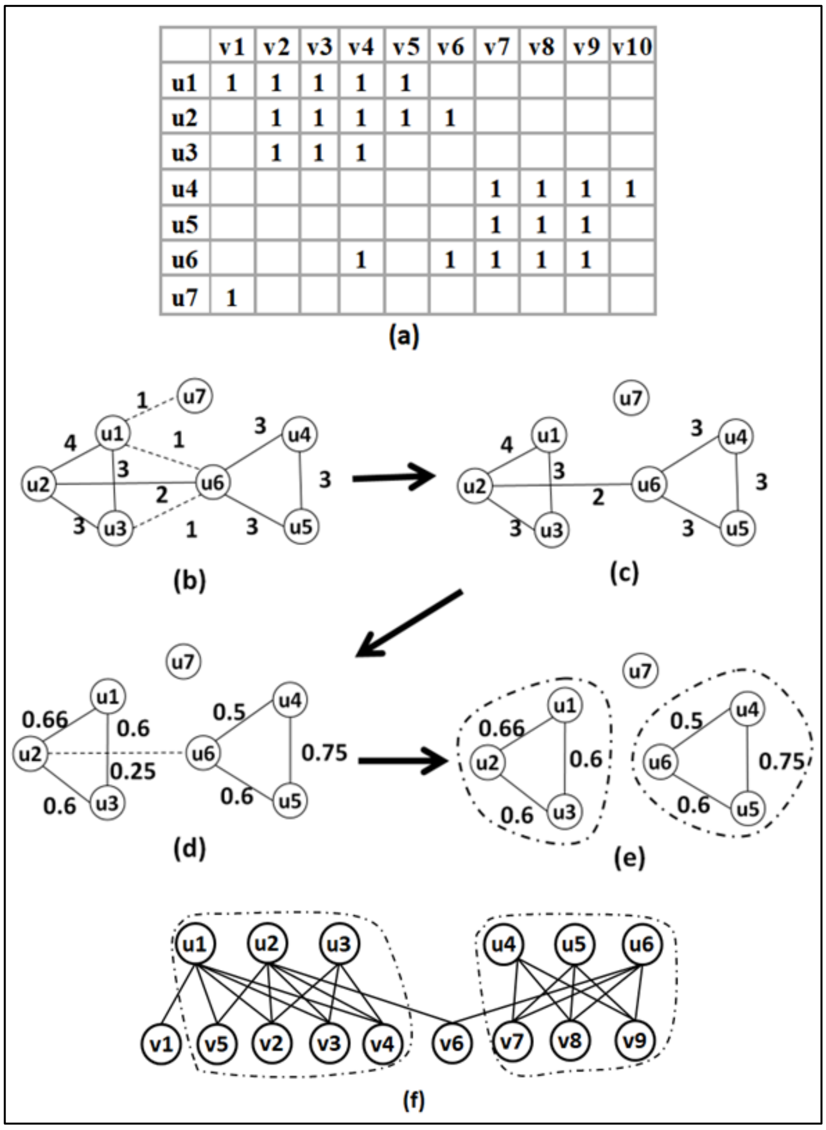
```
1. Construct boolean matrix $F$ such that $F_{i,j} = 1$ when $R_{i,j} \geq Th_{rl}$ and $T_{i,j} \geq Th_{tf}$,
   otherwise $F_{i,j} = 0$
2. For all $i, j$ Construct $G_s$ by adding an edge between $u_i$ and $u_j$ where $F_{ij} = 1$
```

```
3. Apply DPClusO to $G_s$ using parameters $CP$, $CD$, $OV$ and generate cluster
   set $\{C_1, C_2, \ldots C_r\}$ where $C_i \subseteq U$.
4. Find out the neighbor nodes in set $V$ for each cluster from step 3
5. Calculate the probability set $Pvk$ of all neighbor nodes $v \in V$ of $k$th cluster.
6. Select neighbor nodes of $k$th cluster by using condition $Pv_ik \geq Pvth$
7. Construct bicluster by attaching selected neighbor nodes to $k$th cluster
8. Repeat step 5 to 7 for $1 \leq k \leq r$
```

Algorithm 2 constructs a simple graph G_s using user defined threshold values Th_{rl} and Th_{tf} and applies DPClusO clustering algorithm to find out the cluster sets from G_s. Relation number threshold Th_{rl} and Tanimoto coefficient threshold Th_{tf} are used to select the edges of G_s. The values for these thresholds can be calculated based on the characteristics of the input bipartite graph. The DPClusO algorithm is mainly used to cluster a simple graph. DPClusO algorithm takes three input parameters, which are cluster density (*CD*), cluster property(*CP*) and overlapping coefficient (*OV*). The detail of these parameters can be found in [11].

DPClusO algorithm generates overlapping clusters with nodes of set U where each cluster has links to nodes from set V. Let the kth cluster C_k have n nodes, $C_k = \{u_1, u_2, u_3 \ldots\ldots\ldots\ldots u_n\}$ where $u_i \in U$, Neighbor of C_k, that is $N(C_k)$ can be written as $N(C_k)=N(u_1) \cup N(u_2) \cup N(u_3) \ldots\ldots\ldots\ldots\ldots \cup N(u_n)$ where $N(C_k) \subseteq V$. let the total number of distinct nodes in $N(C_k)$ *is* m, such that $N(C_k) = vk = \{v_1, v_2, v_3 \ldots\ldots\ldots\ldots v_m\}$ Step 5 in the Algorithm 2 find out the probability set Pvk of all $v_i \in V$ nodes in k^{th} cluster. We calculate the probability of inclusion of v_i in cluster C_k by using the formula

$$p_{v_ik} = \frac{|N(v_ik)|}{|C_k\,|} \qquad (2.1)$$



	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
u1	1	1	1	1	1					
u2		1	1	1	1	1				
u3		1	1	1						
u4							1	1	1	1
u5							1	1	1	
u6				1		1	1	1	1	
u7	1									

Figure 2. 1 An example demonstrating BiClusO (a) Matrix representation of a bipartite graph (b) Graph construction by relation number (c) Filter by relation number threshold (d) Generate Tanimoto coefficient of edges (e) Filter by Tanimoto threshold and apply DPClusO (f) Second node attachment to each cluster.

Step 6 and 7 in the Algorithm 2 filter out some of the $v \in V$ nodes comparing this probability with the threshold and attach the remaining nodes to the cluster.

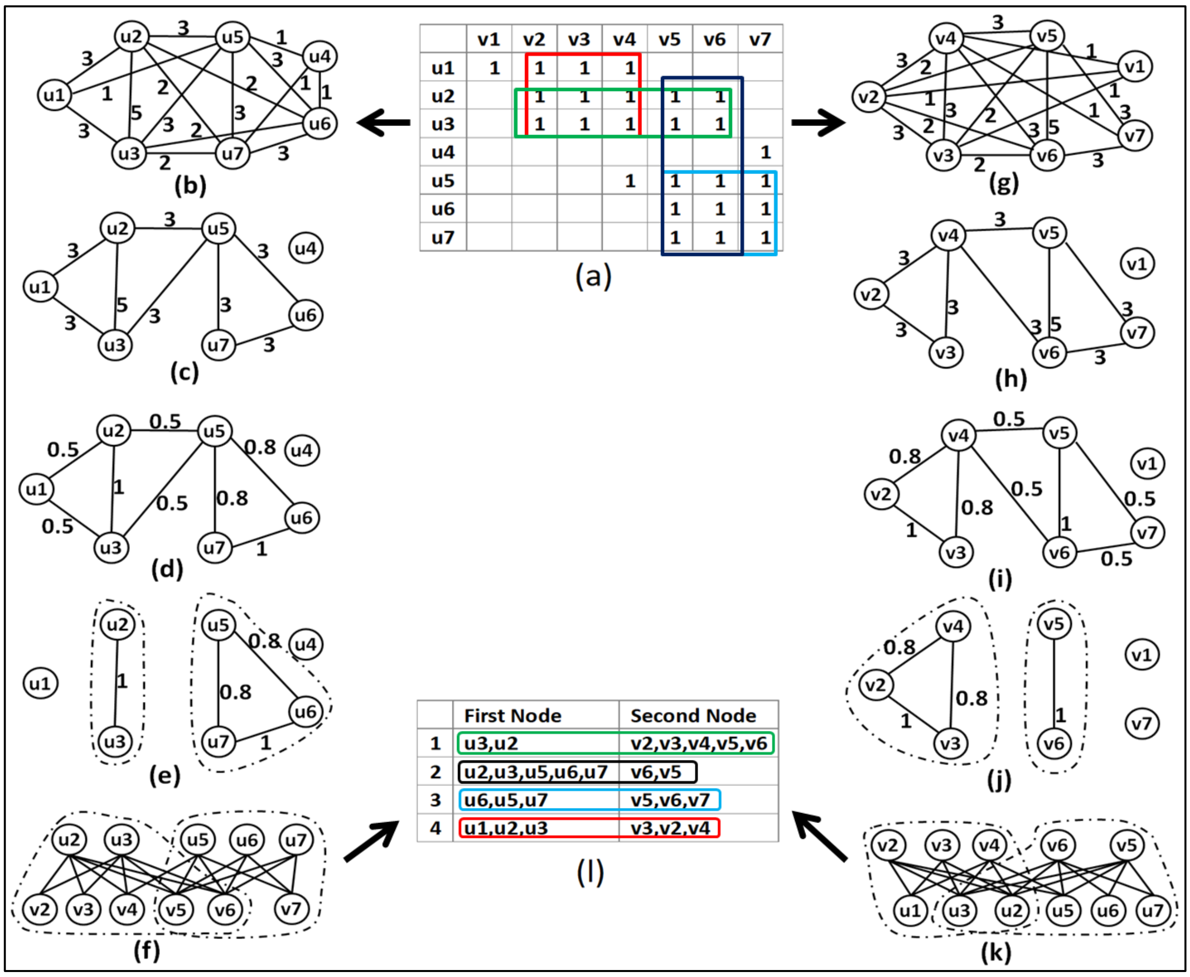
The algorithm can generate bicliques (if exists) by setting *Pvth =1.*

For clear understanding, here we explain our method by an example. Let a simple bipartite graph $G = (U, V, E)$ is represented by the matrix of Fig 2.1(a). Here $|U| =7$, $|V| =10$, *u1* has neighbor set *{v1,v2,v3,v4,v5}* and *u2* has neighbor set *{v2,v3,v4,v5,v6}.* $|N(u1) \cap N(u2)| = 4$ which is relation number i.e. the common neighbor between *u1* and *u2.* The Tanimoto coefficient between these two nodes is $|N(u1) \cap N(u2)| / |N(u1) \cup N(u2)| = 0.66$. The simple graph of Fig 2.1(b) has been constructed by considering the elements of set *U* as nodes and by placing an edge between a node pair when the relation number between them is greater than *0.* From the graph of Fig 2.1(b) we get the graph of Fig 2.1(c) by filtering out the edges corresponding to relation number < *2.* Fig 2.1(d) shows the Tanimoto coefficients corresponding to each edge. Finally, we get the graph of Fig 2.1(e) by filtering out the edges for which Tanimoto coefficient is ≤ *0.25.* Thus both filtering removes the less important edges and improve the possibility of finding good clusters of densely connected subgraphs. By applying DPClusO algorithm to the graph of Fig 2.1(e), we get two clusters *{u1,u2,u3} and {u4,u5,u6}.* Now, to complete the biclusters we add nodes from set V to these clusters using *Pvth > 0.5.* Finally we get two biclusters *{(u1,u2,u3),(v2,v3,v4,v5)}* and *{(u4,u5,u6),(v7,v8, v9)}* as shown in Fig 2.1(f).

2.4 Extended example using both row column wise biclustering

Matrix representation: A simple bipartite graph can be represented by a binary matrix (Fig. 2.2(a)). In a simple bipartite graph, edge weights are confined to 0, 1. For a weighted graph, the edge weights can be converted to binary values by scaling the weights and defining a threshold limit for 0 and 1. In Fig. 2.2(a), the row and column labels belong the sets of *U* and *V* respectively. We place 1 to each cell of the matrix if there exists an edge between the corresponding row label and column label in the original graph

Simple graph Construction: Fig 1(a) denotes a binary matrix of a simple bipartite graph where $|U| = 7$, $|V| = 7$. We used data folding mechanism to generate two independent simple graphs involving the elements of U and V



	v1	v2	v3	v4	v5	v6	v7
u1	1	1	1	1			
u2		1	1	1	1	1	
u3		1	1	1	1	1	
u4							1
u5				1	1	1	1
u6					1	1	1
u7					1	1	1

	First Node	Second Node
1	u3,u2	v2,v3,v4,v5,v6
2	u2,u3,u5,u6,u7	v6,v5
3	u6,u5,u7	v5,v6,v7
4	u1,u2,u3	v3,v2,v4

Figure 2. 2 Biclustering method; (a) matrix representation (b)-(e)simple graph construction (row wise data folding); (g)-(j)simple graph construction (column wise data folding); (e),(j)applying DPClusO; (f),(k) second node attachment; (l) join cluster set.

separately. In the weighted graph of Fig. 2.2(b), the edge weights are determined based on common neighbors between corresponding nodes in the original bipartite graph which we call relation number. For example, node *u1* has neighbor set *{v1, v2, v3, v4}* and *u2* has neighbor set *{v2, v3, v4, v5, v6}*. So the relation number between *u1* and *u2* is $|N(u1) \cap N(u2)| = 3$. From Fig. 2.2(b) some of the edges are filtered out using the relation number threshold < 3 which generate the graph of Fig. 2.2(c). Then Tanimoto coefficient is calculated for each edge. For example Tanimoto coefficient between node *u1* and *u2* is $|N(u1) \cap N(u2)| / |N(u1) \cup N(u2)| = 0.5$. Fig. 2.2(d) shows the Tanimoto coefficients for all the edges. Tanimoto coefficient of an edge measures the robustness of relation between the nodes it connects. From Fig. 2.2(d) some of the edges have been filtered out by setting the Tanimoto coefficient threshold > 0.5, which makes the simple graph of Fig. 2.2(e). By following the same way, the simple graph of Fig. 2.2(j) has been created involving the elements of set *V*. Applying DPclusO: After applying both types of filtering the remaining graph shows a significant amount of the reduction of less important edges. DPClusO[20][21], which was developed by extending the concepts of DPClus algorithm [22][23], can easily separate clusters from such simple graphs. The overlapping controlling parameter of DPClusO can be used to generate overlapping clusters to some extent. The dotted line in Fig. 2.2(e) and (j) shows the separable region of the clusters.

Second node attachment: After separating the clusters *(u2, u3),(u5, u6, u7),(v5, v6)* and *(v2, v3, v4)* as shown in Fig. 2.2(e) and (j), the corresponding neighbor of each element of the clusters are joined by using their attachment probability. The attachment probability of each neighbor node in a cluster is calculated by dividing the number of corresponding cluster nodes attached to it by the total number of nodes in the cluster. After calculating the attachment probability, some of the neighbor nodes are pruned by using attachment probability threshold. We can extract the bicliques by setting the attachment probability to be 1. In this above example we consider the attachment probability threshold > 0.5. After attachment we get four biclusters *{(u2, u3),(v2, v3, v4, v5, v6)}*, *{(u5, u6, u7),(v5, v6, v7)}*, *{(v5, v6),(u2, u3, u5, u6, u7)}*, and *{(v2, v3, v4),(u1, u2, u3)}* as shown in Fig. 2.2(f) and (k).

Join cluster sets: Finally, four biclusters are listed in Fig. 2.2(l) by arranging the nodes from *U* and *V* as first and second set respectively. Sometimes finding biclusters from both sides create some duplicate biclusters or similar clusters with big overlapping. This duplicity can be removed by keeping only one set from each duplicate pair. We also introduced the biclustering overlapping coefficient which can be used to join or filter out similar biclusters to some extent.

2.5 Overlapping parameter

We used here two different overlapping coefficients which are 1) simple graph cluster overlapping coefficient and 2) bicluster overlapping coefficient.

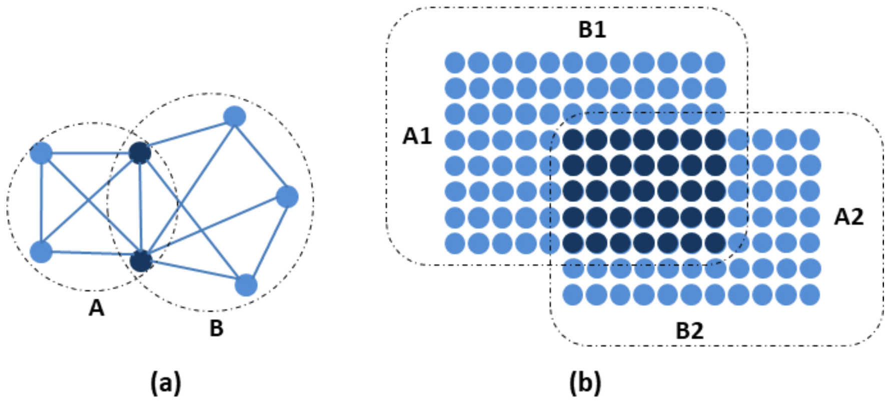


Figure 2. 3 a) Simple cluster overlapping coefficient b) bicluster overlapping coefficient.

2.5.1 Simple overlapping parameter

The first overlapping coefficient measures the overlapping between two clusters while applying DPClusO to the simple graph generated from the bipartite graph. If A, B ae two sets simple clusters (Fig 2.3 (a)) and $A \cap B$ be the set of common number of nodes between them then the simple cluster overlapping coefficient is

$$SCov = \frac{|A \cap B|^2}{|A| \times |B|} \qquad (2.2)$$

2.5.2 Bicluster overlapping parameter

Bicluster overlapping coefficient measures the overlapping between two biclusters. This measurement considers each bicluster as a biclique and finds the common block matrix in terms of total block area occupied by them. If the node sets two biclusters are denoted by *A1, B1* and *A2, B2* (Fig 2.3(b)). The block area which denotes the number of edges in each bicluster is $|A1| \times |B1|$ and $|A2| \times |B2|$. The common block area between two biclusters is $|A1 \cap A2| \times |B1 \cap B2|$. The overlapping coefficient can be expressed by the following Equation

$$BCov = \frac{|A1 \cap A2| \times |B1 \cap B2|}{|A1| \times |B1| + |A2| \times |B2| - |A1 \cap A2| \times |B1 \cap B2|} \tag{2.3}$$

2.6 Computational complexity

A bipartite graph is a graph that consists of two disjoint sets of nodes, U and V, such that each edge connects a node in U with a node in V, i.e., U and V are independent sets. Let $|U| = n$ and $|V| = m$ and therefore, the bipartite graph can be represented by a matrix of size $n \times m$. The detail of the algorithm of BiClusO is available in [24]. In BiClusO, calculating relation numbers between $n(n-1)/2$ pairs of the nodes of set U requires $mn(n-1)/2$ computations (Fig. 1(b)). Therefore, the complexity for this calculation is $O(n^2m)$. Similarly, the complexity for calculating Tanimito coefficient is $O(n^2m)$ (Fig 2.2(d)). Filtering the edges in the weighted graph and construction of a simple graph using the threshold values of the Tanimoto coefficient and relation number need $n(n-1)/2$ or $O(n^2)$ computations (Fig. 2.2(c) and 2.2(d). The computational complexity for DPClusO algorithm is $O(n^3)$ where n is the total number of nodes in a simple graph [11][12] (Fig. 2.2(e)). If the clusters generated by DPClusO on the simple graph are of sizes *p1, p2...pc*, then assigning the second node set to all the clusters needs *p1m, p2m....pcm = (p1 + p2....pc)m* computations. The value of *p1 + p2....pc* is roughly equal to n. Thus the computational complexity for assigning the second node by calculating the attachment probability is $O(nm)$. So the total computational complexity of biclustering on the first node set comprises of 1) weight graph construction using relation number and Tanimoto coefficient 2) filtering and construction of simple graph 3) simple graph clustering and 4) second node attachment which is in total $O(2n^2m + n^2 + n^3 + nm)$. Similarly, the computational complexity considering the second node set is $O(2m^2n + m^2 + m^3 + mn)$.

2.7 Conclusion

We developed a novel biclustering algorithm called BiClusO based on a data folding mechanism and the DPClusO algorithm previously developed by our group. We explain the algorithm using row-wise or both row and column wise. The two different overlapping parameters help to control the overlapping property among biclusters. Using this property user can generate a maximum or a minimum number of bicluster set from a bipartite graph.

Chapter 3

Comparision of BiClusO with different biclustering algorithms

We compare our algorithm with several other biclustering algorithms in the context of two different types of biological datasets and four synthetic datasets with known embedded biclusters. Biclustering technique is widely used in different fields of studies for analyzing bipartite relationship dataset. Over the past decade, different biclustering algorithms have been proposed by researchers which are mainly used for biological data analysis. The performance of these algorithms differs depending on dataset size, pattern, and property. These issues create difficulties for a researcher to take the right decision for selecting a good biclustering algorithm. Two different scoring methods along with Gene Ontology(GO) term enrichment analysis have been used to measure and compare the performance of our algorithm. Our algorithm shows the best performance over some other well-known biclustering algorithms.

3.1 Introduction

Different biclustering algorithms have been implemented which usually produce the textual presentation of output and are widely used mainly for analyzing biological data [14][18]-[23][25][26]. Most of these implementations do not have overlapping controlling parameter or technique to separate bicliques from bicluster set. Moreover, only a few algorithms have been implemented with comprehensive visual presentations of biclusters[27]-[30]. Researchers have compared the performance of these biclustering algorithms by using artificial data and real biological data [26][31][32]. Due to performance variations mentioned in literature, it is very difficult to make the right decision to select an efficient biclustering algorithm. Optimal parameter selection for these algorithms is another important issue which needs persistent trial. We selected five different biclustering algorithms BiMax, Spectral, CC, Plaid and xMOTIF for comparing with our algorithm. We selected these algorithms based on three criteria 1) whether the implementation is available in R 2) whether the algorithm can deal with binary dataset 3) whether the algorithm can produce some reasonable number of biclusters(at least two). Implementation of these algorithms is available within R package 'biclust' [33]. We perform the comparison using three different types of datasets i.e. species-VOC (volatile organic compound) relational data, gene-expression data of C. Cerevisiae and synthetic data. We created synthetic datasets of known embedded biclusters of varying size and overlapping properties. In the current work, we report the extended results of the experiment with another biological dataset and on the implementation of the algorithm using GUI.

3.2 Reference methods and dataset selection

This section illustrates the reference biclustering algorithms, data selection and data preparation for evaluating the performance of BiClusO.

3.2.1 Reference BiClustering algorithms

We have chosen five other well-known biclustering algorithms namely BiMax, Plaid, xMOTIF, Spectral, And CC (Chang and Church) for performance comparison with BiClusO. These algorithms have been implemented in the R package 'biclus' which we utilized for our experiments. BiMax recursively divides the binary data matrix using a different concentration of regions made by 1 and 0 and generates all maximal biclusters [26]. It can only work with binary data matrix. Plaid algorithm models the data matrices with the sum of different layers and an assumed number of biclusters. It fits the model by iteratively updating the different parameters [25]. xMOTIF discretizes the data matrix by searching a set of rows following the same linear order under a set of columns to find motif [20]. Spectral algorithm uses the singular value decomposition in eigenvectors to search the coherent value over row and column where the variance is lower than a given threshold value [21]. CC uses the deterministic greedy method based on the mean square residue of a submatrix where the score is lower than a threshold [14].

3.2.2 Selection of dataset

We used two different types of datasets i.e. biological data and synthetic data. It is easy to compare the performance of different biclustering algorithms when the properties of the results are foreknown. Two different biological datasets are speciesVOC (volatile organic compound) data and microarray gene expression data of S. Cerevisiae. We evaluated the performance of different algorithms by measuring the statistical significance of the generated biclusters in terms of biological properties. We also created synthetic dataset by embedding biclusters with different physical properties. These physical properties help us to measure the quality of the extracted biclusters by different algorithms.

Species-VOC data

The species-VOC bipartite dataset was collected from KNApSAcK [15]-[17] database. This dataset forms a sparse matrix of dimension 710 species vs. 1760 VOCs emitted by those species under different biological stress. We categorized these species under five different taxonomic levels which are kingdom, phylum, class, order, and family. Usually, similar species group under family level produce many distinct types of VOCs[34]. However, there are some common VOCs among such species groups.

Gene expression data

This dataset is the gene expression data of the species S. Cerevisiae [35]-[39] consisting of a matrix with dimension 2644 genes vs. 79 conditions. The data point in each cell represents the logarithmic ratio of the expression levels of a particular gene under two different experimental conditions. Each column represents n logtransformed expression-level ratios of n genes for a single chip. We digitized the data by column-wise transformation using the following formula

$$D_{i,j} = \begin{cases} 1, & M_{i,j} \geq avg_j + th \times sd_j \ \ or \ \ M_{i,j} \leq avg_j - th \times sd_j \\ 0, & otherwise \end{cases} \qquad 3.1$$

Where *avg*$_j$ and *sd*$_j$ represent average and standard deviation of the jth column. The threshold *th* is the value which determines the quantity of 1s to be set in the digital matrix.

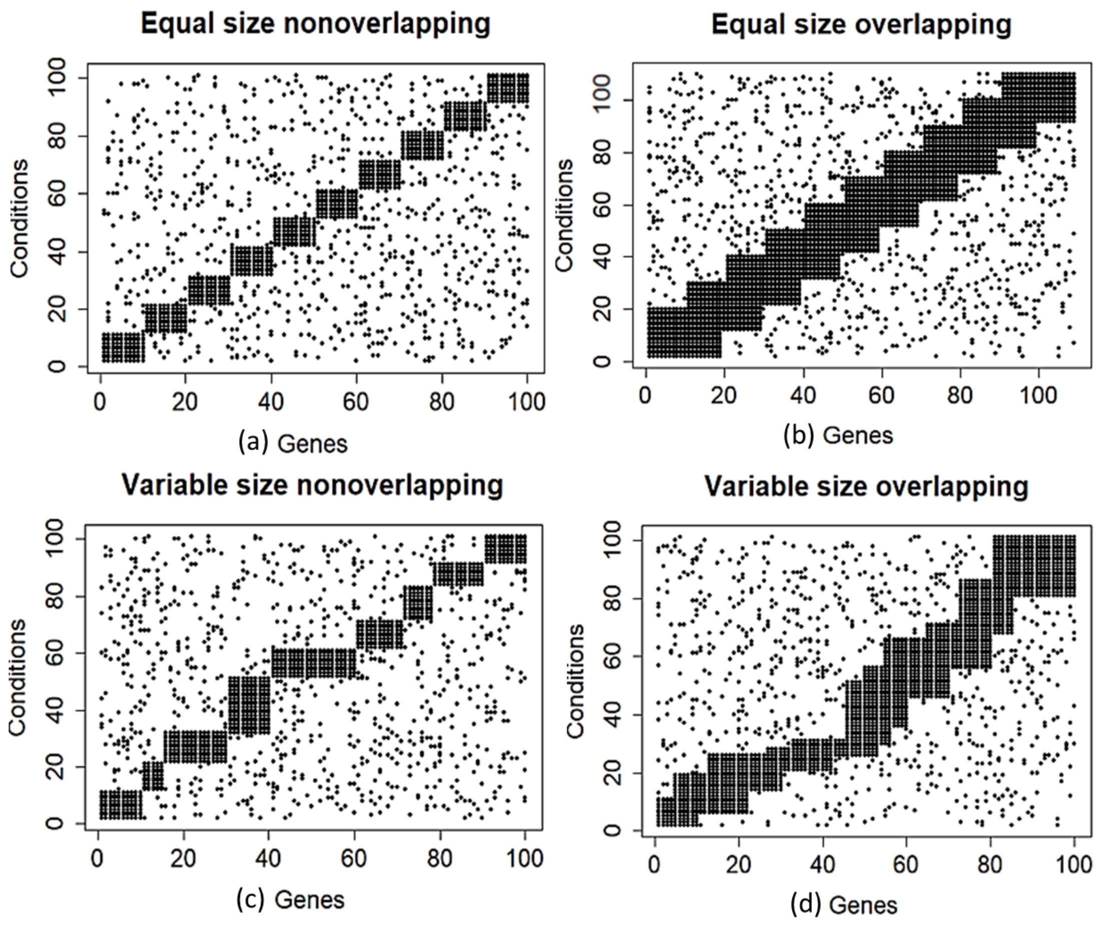


Figure 3. 1 Synthetic dataset with maximum noise rate.

After accessing three different threshold values (th = 0.5, 1 and 1.5) we considered the th to be 1.5 which makes the digitized matrix as a sparse matrix.

Synthetic data

We created synthetic dataset of binary matrices using hypothetical relations between genes and conditions. We inserted a block of 1s as embedded bicluster to such matrix where 1 represents the differential expression of a gene under a condition. We changed the overlapping property and size of the biclusters and constructed four different types of such dataset: (1) equal size nonoverlapping, (ii) different size nonoverlapping, (iii) equal size overlapping and (iv) different size overlapping. Each dataset is a matrix of dimension 100 genes vs. 100 conditions. Each bicluster of equal size nonoverlapping set has 10 genes and 10 conditions (Fig. 3.1(a)). Different size nonoverlapping set has a varying number of genes between 5 to 10 and a varying number of conditions between 8 to 20 (Fig. 3.1(c)). Each bicluster of equal size with overlapping consists of 18 genes and 18 conditions where 9 genes and 9 conditions are overlapped between biclusters (Fig. 3.1(b)). Each bicluster of different size with overlapping consist of maximum 18 genes and 18 conditions with the overlapping region varying between 1 to 8 for genes and 2 to 10 for conditions (Fig. 3.1(d)). We introduced noise to the non-cluster region of each dataset by randomly inserting '1's. Total 36 datasets were created with four types of variation and nine different noise levels from 1% to 9%.

3.3 Parameter Setting

We used the default parameter setting recommended by different papers and the authors of the biclustering algorithms [26][31][32]. In some cases, we slightly changed the parameters to force the algorithm to generate at least a minimum number of biclusters. For BiMax we set minimum row = 2, and minimum column=2. For Spectral we set minimum row =2, minimum column =2, maximum within variation =1, normalization = 'log' and number of eigenvalue = 1. A high number of eigenvalue for this algorithm creates a performance issue on calculation time. The recommended value is 1. For xMOTIF we set *sd*=7, *alpha*=0.1, *ns*=10, *nd*=1000. For CC we set *alpha*=1.2, *delta* = .0000005. We took delta to such small value to create at least two biclusters. We set default parameter for Plaid algorithm of R BiClust package. For our algorithm we set *CD*(Cluster density)=0.5, *CP*=0.5(Cluster property), Tanimoto coefficient = 0.33, relation number = 5 and attachment probability = 0.5 Cluster density and Cluster property are default parameters for DPClusO algorithm. From our previous experience on some actual dataset, the optimal setting for both CD and CP is 0.5 [40][41] which produced good results. Also user can change the cluster density between 0.5 to 0.7. Most of the cases of the actual dataset are sparse by nature. To select the best threshold on sparse matrix user can start with relation number = 2 or 3 and Tanimoto coefficient = 0.33. Tanimoto coefficient $\geq$ 0.33 allows more than 50% similarity between 1s of two binary vectors. Very high relation number and Tanimoto coefficient might exclude some nodes from the analysis. If the data is not sparse then Tanimoto coefficient threshold can be adjusted between 0.4 to 1.0 depending on whether the required number of nodes are included in the biclusters.

3.4 Results of performance evaluation

We adopted different scoring methods to evaluate the strength of biclustering algorithms. The results of comparison based on different biological data and synthetic data are summarized in Fig. 3.2, Fig. 3.3, Fig 3.4 and Fig. 3.5. In the following we discuss in detail.

3.4.1 Classification of species based on VOCs

Biclustering can be used to classify species based on the VOCs they emit. We applied the hypergeometric test to the bicluster set generated from species-VOC relational data. The richness of same categorical species in a bicluster in terms of family-level taxonomy was evaluated by using the following formula

$$p\ value = 1 - \sum_{i=0}^{n-1} \frac{\binom{G}{i}\binom{N-G}{X-i}}{\binom{N}{X}} \qquad 3.2$$

Here X is the number of species in a bicluster. N is the number of species in the data. n is the maximum number of same categorical species belong to a specific family in a cluster. G is the total number of species of that category in the data. After selecting the statistically significant biclusters where $p\text{–}value \leq 0.05$, we combined the results generated by different algorithms. We created a list containing algorithm name and their corresponding *–log(p–value)* in each row. We rearranged the list by descending order and therefore the top row on the list indicates the significant bicluster and the bottom row indicates the least significant bicluster out of all the biclusters generated by all the algorithms.

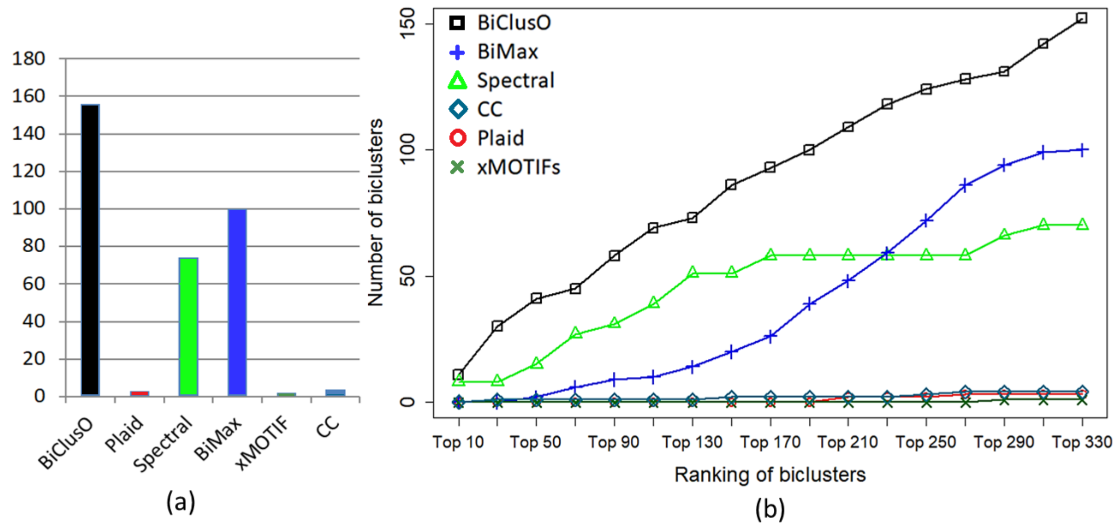


Figure 3. 2 a) Number of biclusters generated by different algorithms. b) Performance of different algorithm on Species VOC data set.

In order to compare the performance of different algorithms, we selected different sets of rows from the top of the list and counted the number of rows corresponding to the different algorithms within each set. The number of biclusters generated by BiClusO, BiMax, and Spectral is more compared to other algorithms as shown in Fig. 3.2(a). Except some slightly overlapping, the BiClusO produces almost distinct biclusters. Plaid, CC and xMOTIF produce a very small number of biclusters which implies that these algorithms are not suitable for finding biclusters from sparse matrices [42]. One of the limitations of BiMax is that it requires to specify the number of biclusters beforehand, otherwise produces 100 biclusters by default. Fig. 3.2(b) shows that BiClusO has the highest share among top ranking clusters based on statistical significance. Spectral biclustering shows the second best performance. A moderate number of the biclusters generated by Spectral has identical sets of species over different sets of VOCs.

3.4.2 Richness of similar function genes

Biclustering of gene expression data is expected to accumulate similar function genes in individual biclusters. In order to find the richness of similar function genes in the gene side of each bicluster, we calculated hypergeometric p-values corresponding to three different Gene Ontology (GO) terms i.e. Biological Process(BP), Cellular Component(CC), Molecular Function(MF). We used the GOstats package from R for calculating the p − values. The result of this analysis produces a series of p−value related to different GO terms. The GO term for which the p−value is the smallest is the most significant. For each bicluster, we selected the lowest p−values for certain term under each type of ontology. The number of generated biclusters by different algorithms is different. For the sake of fair comparison, we took up to the best 50 biclusters from each algorithm. For the algorithms that produced less than 50 clusters, we took all of them. We combined the selected clusters of all algorithms corresponding to a specific ontology and sorted the clusters according to their respective –log(p − value).

Scoring results with respect to three different GO terms are summarized in Fig. 3.3.

Five biclustering algorithms including BiClusO produced some meaningful biclusters in terms of biological significance. We tried with the spectral algorithm by changing the reference parameter to produce some reasonable number of biclusters but failed. Only BiClusO and BiMax produced a significant number of biclusters with the small number of nodes in both gene and condition sides. Plaid, xMOTIF, and CC (algorithm) produced a small number of biclusters with a large number of nodes

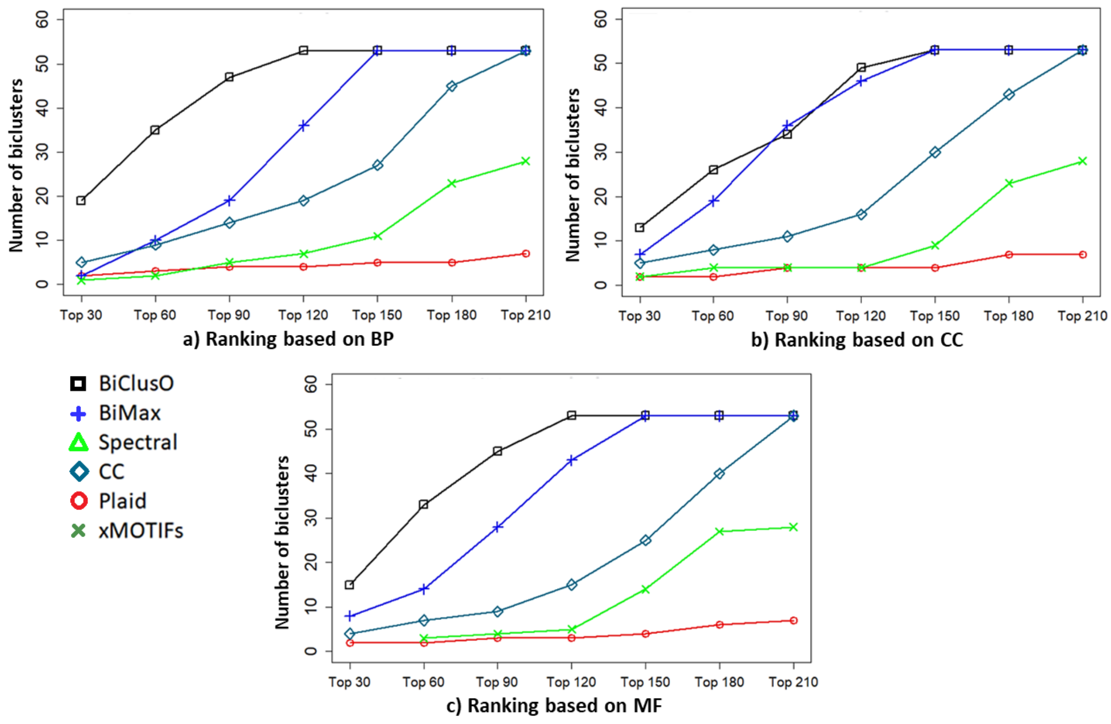


Figure 3. 3 Performance of different algorithm based on gene expression data of S. cerevisiae.

in both gene and condition sides. According to Fig. 3.3 in all three cases of BP, CC and MF, BiClusO produces most of the best ranking biclusters among the first 30 top biclusters. In the case of BP and MF (Fig. 3.3(a) and Fig. 3.3(b)), BiClusO clearly outperformed the other algorithms. Only in case of CC, BiMax shows almost similar performance like BiClusO. The CC (algorithm) shows the third position in all three ontology analysis. Plaid produces the smallest number of biclusters with large sizes compare with other algorithms which fail to produce good p-value.

3.4.3 Comparison based on synthetic data

We evaluated the degree of similarity of bicluster sets generated by an algorithm from synthetic data by using the following matching score formula [26][31].

$$S_g(A_1, B_1) = \frac{1}{\lceil A_1 \rceil} \sum_{(G_1,C_1)\in A_1} max_{(G_2,C_2)\in B_1} \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|} \qquad 3.3$$

Here *A1* denotes the actual bicluster set and *B1* denotes the bicluster set generated by an algorithm. An element of the bicluster set *A1* is denoted by *(G1, C1)* where *G1* is the set of genes and *C1* is the set of conditions. Similarly *(G2, C2)* denotes an element of the bicluster set *B1*. The Jaccard index formula: | *G1 ∩ G2* | / | *G1 ∪ G2* | express the similarity between two biclusters in terms of gene side. According to the Jaccard index, identical biclusters produce maximum value 1 whereas disjoint biclusters produce minimum value 0. The above equation finds the average of maximum matching score of each gene set from *A1* to *B1*. Similarly, the matching score *S*$_c$*(A1, B1)* was calculated considering the matching of condition sides of biclusters. Finally the formula

$$S(A_1, B_1) = \sqrt{S_g(A_1, B_1) * S_c(A_1, B_1)} \qquad 3.4$$

calculates the overall matching score from *A1* to *B1* considering both genes and conditions. If *A1* be the actual bicluster set and *B1* be the generated bicluster set by an algorithm then the matching score *S(A1, B1)* represents average module recovery whereas *S(B1, A1)* represents average cluster relevance[26]. Average module recovery reflects the algorithm's ability to retrieve the actual biclusters. The best case value for this score is 1 which means that all of the actual biclusters are successfully discovered by the algorithm. The number of biclusters generated by the algorithm, in this case, must be greater than or equal to the number of actual biclusters. Average cluster relevance measures the similarity between generated bicluster and actual bicluster. The best case value for this score is 1 which means maximum similarity is achieved by the algorithm.

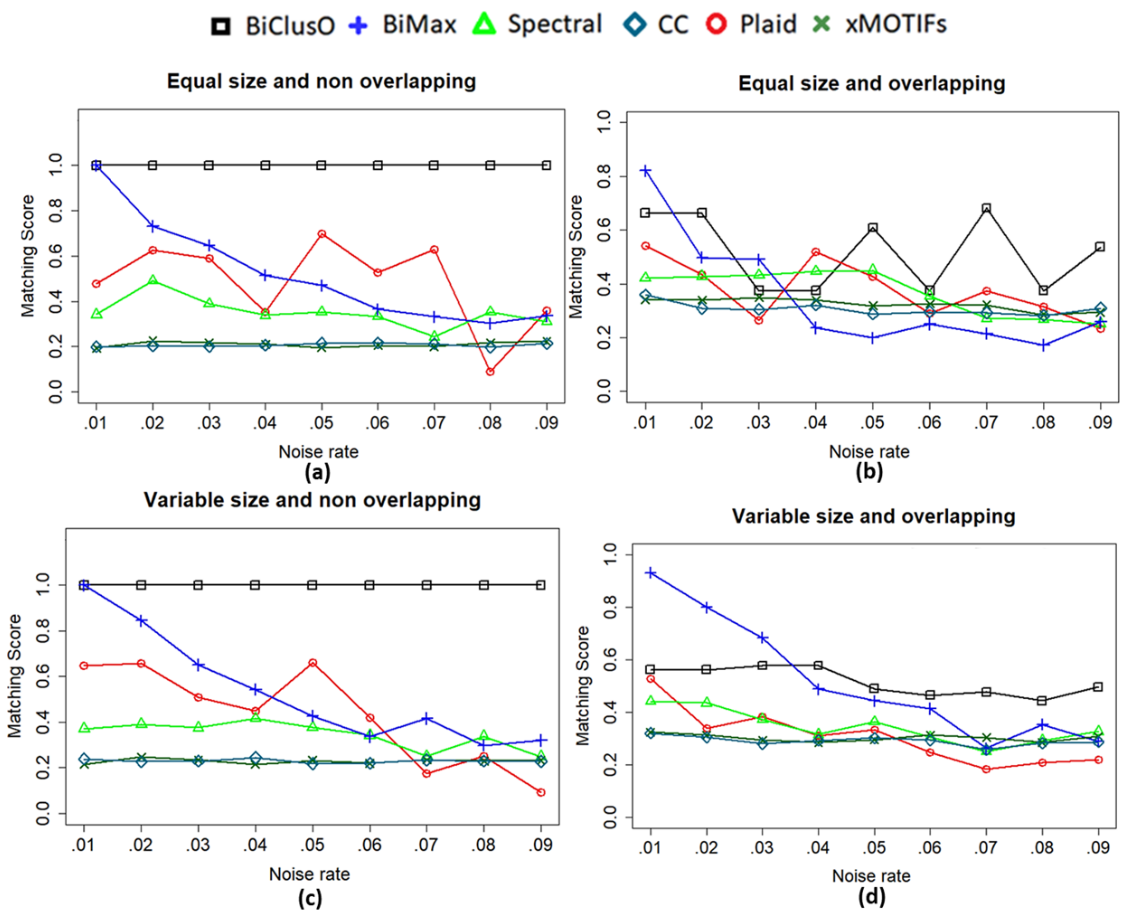


Figure 3. 4 Performance of different algorithms in terms of average module recovery on synthetic data set.

The number of biclusters generated by the algorithm, in this case, must be less than or equal to the number of actual biclusters. If both scores are 1 then the algorithm successfully generates actual biclusters.

Performance on average module recovery

Fig 3.4 shows the performance of average module recovery for all algorithms. BiClusO outperforms over other biclustering algorithms on the synthetic dataset of equal size nonoverlapping (Fig. 3.4(a)) and variable size nonoverlapping (Fig. 3.4(c)). On both dataset maximum matching score, '1' is achieved over different noise level.

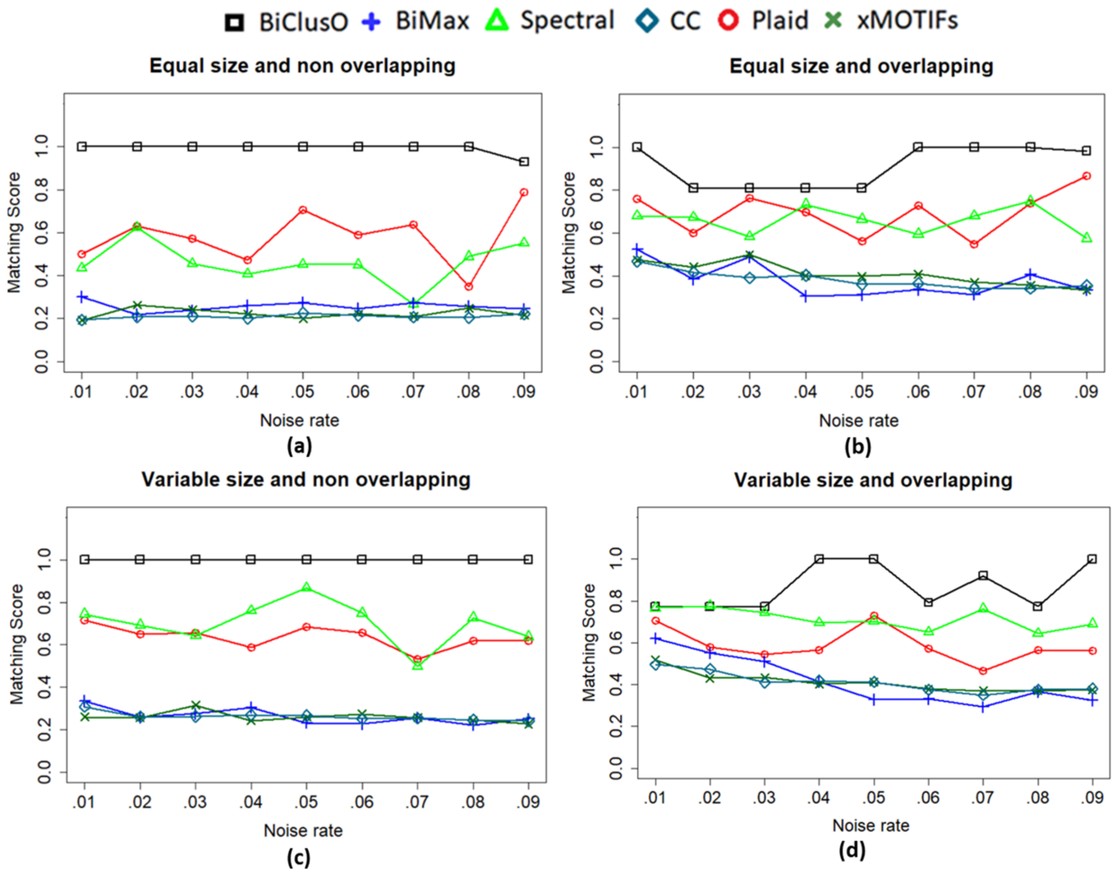


Figure 3. 5 Performance of different algorithms in terms of average cluster relevance on synthetic data set.

BiMax and Spectral achieve the second and third position. BiClusO shows good performance over BiMax and Spectral in Fig. 3.4(b) of equal size overlapping data. From Fig. 3.4(d) of variable size overlapping data, BiMax performs better matching score while the noise level is low but as the noise level increases the performance drastically degrades. Most of the cases of four types of synthetic dataset with a varied level of noises, BiClusO achieves the best results in terms of average module recovery.

Performance on average cluster relevance

Almost all of the cases of four different synthetic datasets BiClusO shows the best performance over other algorithms in terms of average cluster relevance (Fig 3.5). Spectral and Plaid achieve the second highest position alternatively. BiMax, xMOTIF and CC show poor performance. The Performance of BiMax is deteriorated because it produces a large number of biclusters where a substantial portion of them are dissimilar to the original bicluster set. Most of the biclusters produced by BiClusO show maximum similarity to the original bicluster set.

3.5 Conclusion

In this study, we present a GUI based implementation of our algorithm BiClusO and compare its performance with different biclustering algorithms based on two types of biological data and four types of synthetic data. Biclusters generated based on biological dataset are analyzed on the basis of statistical significance determined by Hypergeometric test in the context of taxonomical classes and different GO terms. The varying properties of size and overlapping on synthetic datasets simulate a diverse pattern of gene-condition bipartite dataset which helps to compare different algorithms effectively. The implementation allows us to control the overlapping between biclusters. Also the attachment probability parameter helps to extract biclusters as bicliques. The performance of our algorithm in terms of different scoring methods on real biological datasets expresses the robustness and effectiveness of BiClusO over other algorithms. The consistency of the performance of different algorithms to a certain degree in the case of synthetic datasets implies the correctness and significance of the comparison method we adopted in the present work

Chapter 4

Application of BiClusO to species-VOC bipartite data

Volatile Organic Compounds are organic compounds that can easily become vapor under environmental temperature and pressure. Most of these VOC contain carbon along with hydrogen, oxygen, chlorine, fluorine, bromine, sulfur or nitrogen. In this work, we mainly focus on VOCs that are emitted by living organisms which are called biogenic VOCs. They have huge environmental and ecological impact due to mutual interaction between species. VOCs play an import role on controlling the ecosystem with individual and combined effect to energize the survival of species. VOC emission is also substantially affected due to impact of climate change and ecosystem redistribution.

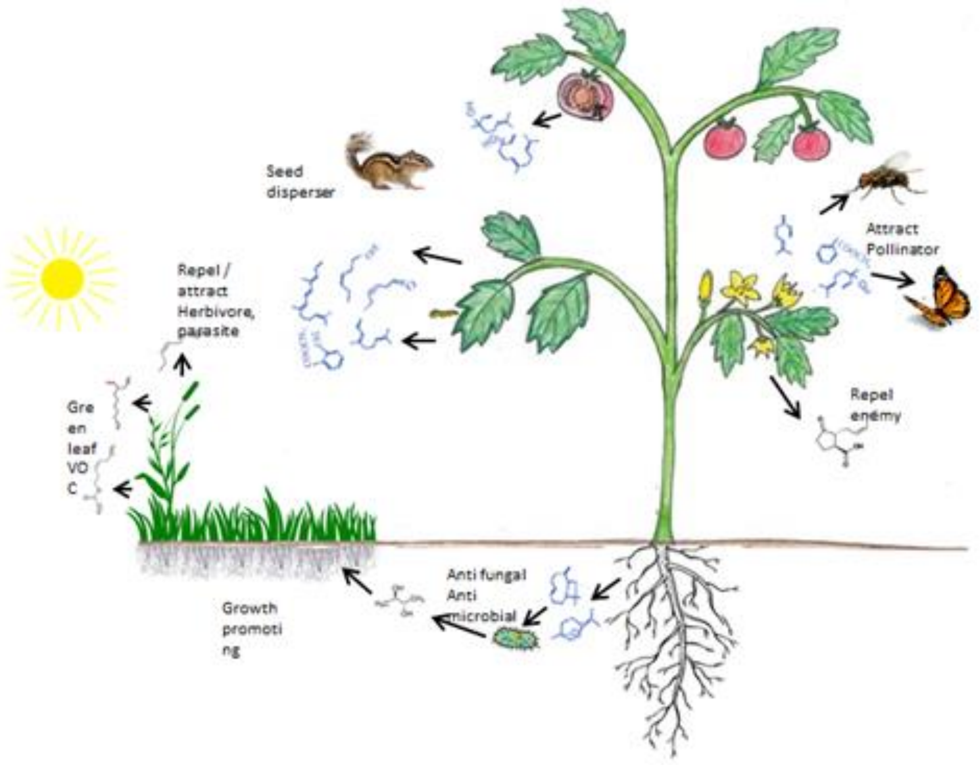


Figure 4. 1 Ecology of volatile organic compound.

Evolutionary development of species partially depends on these changes of environment and mutual interaction of species by using VOCs. Backtracking VOC generation as a product of intricate metabolic pathways can identify complex cellular processes, which can then be used to categorize the biosynthesis mechanisms of these metabolites in different species. Having successful discovery of some VOC pathways, scientist are now trying to control the floral VOCs of plants through metabolic engineering such as upregulating or down-regulating of biochemical steps and modification of existing pathways in an attempt to increase pollination and defense mechanisms [43].

4.1 Introduction

Classification of species based on common emitting VOCs can lead to understanding the symbiosis of organisms in terms of VOCs. Also, clustering of VOCs based on chemical structure similarities can help to predict their pathways. In this study, we applied our algorithm to cluster the species-VOC relational data from the KNApSAcK database [15]- [17] and determined that the VOC-based classification of species was consistent with the taxonomy-based classification. We also assessed the diversity of VOC pathways across different kingdoms of species.

4.2 Data Collection and Preprocessing

We collected species-VOC relationship data from the KNApSAcK metabolite ecology section of the KNApSAcK family databases [17]. We also collected additional data from different papers and journals using Google Scholar and other publication sites [44] - [74]. The final data consists of 12410 species-VOC relations including 710 species and 1740 different VOCs. We applied our proposed biclustering algorithm to the species-VOC relationship data. As shown in Table 4.1, taxonomically, the species are from five different kingdoms. Most of the species belong to the plant, fungi, or bacteria kingdoms. In most of the literatures, VOCs are described by chemical names. Sometimes a VOC is identified by different names according to different

Table 4. 1 VOC data according to different taxonomy.

Kingdom	Phylum	Class	Order	Family	Total	Voc
Bacteria	11	4	43	60	420	773
Eubacteria	1	1	1	1	1	1
Euryarchaeota	1	1	1	1	2	5
Fungi	4	16	22	34	148	598
Plantae	3	5	7	16	139	793

chemical naming conventions. For example, the sweet scent compound *a-humulene* has four different names; *alpha-humulene*; *humulene*; *alphacaryophyllene*; and *3,7,10-humulatriene*. For our study, we selected only one name for each compound. In total, the 710 different species with 1740 distinct VOCs have many-to-many relationships which forms a big bipartite graph. The number of reported VOCs emitted by a species varied between 1 and 157. Fig. 4.2(a) shows the frequency of species with respect to the reported number of emitted VOCs based on our database. The uneven distribution of VOCs is due to the fact that some research emphasizes retention time or percentage volume of a small number of emitted VOCs over the complete set of emitted VOCs. Some experiments emphasize an important set of VOCs which have specific activities associated with characteristics such as plant growth, pollinator attraction, resisting enemies, or disease biomarkers. Only some literature reports the complete VOC profiles of species. From Fig. 4.2(a), we can see that 122 species are associated with only one VOC, 51 species with only two VOCs, 50 species with only three VOCs, and so on. For our purposes, we completely discarded those species which are associated with only one or two VOCs. Finally, we produced an input bipartite graph G with dimension 540 x 1710.

4.3 Threshold selection and biclustering

In step three of algorithm 2 from chapter 2, the DPClusO algorithm generates a cluster set using three parameters.

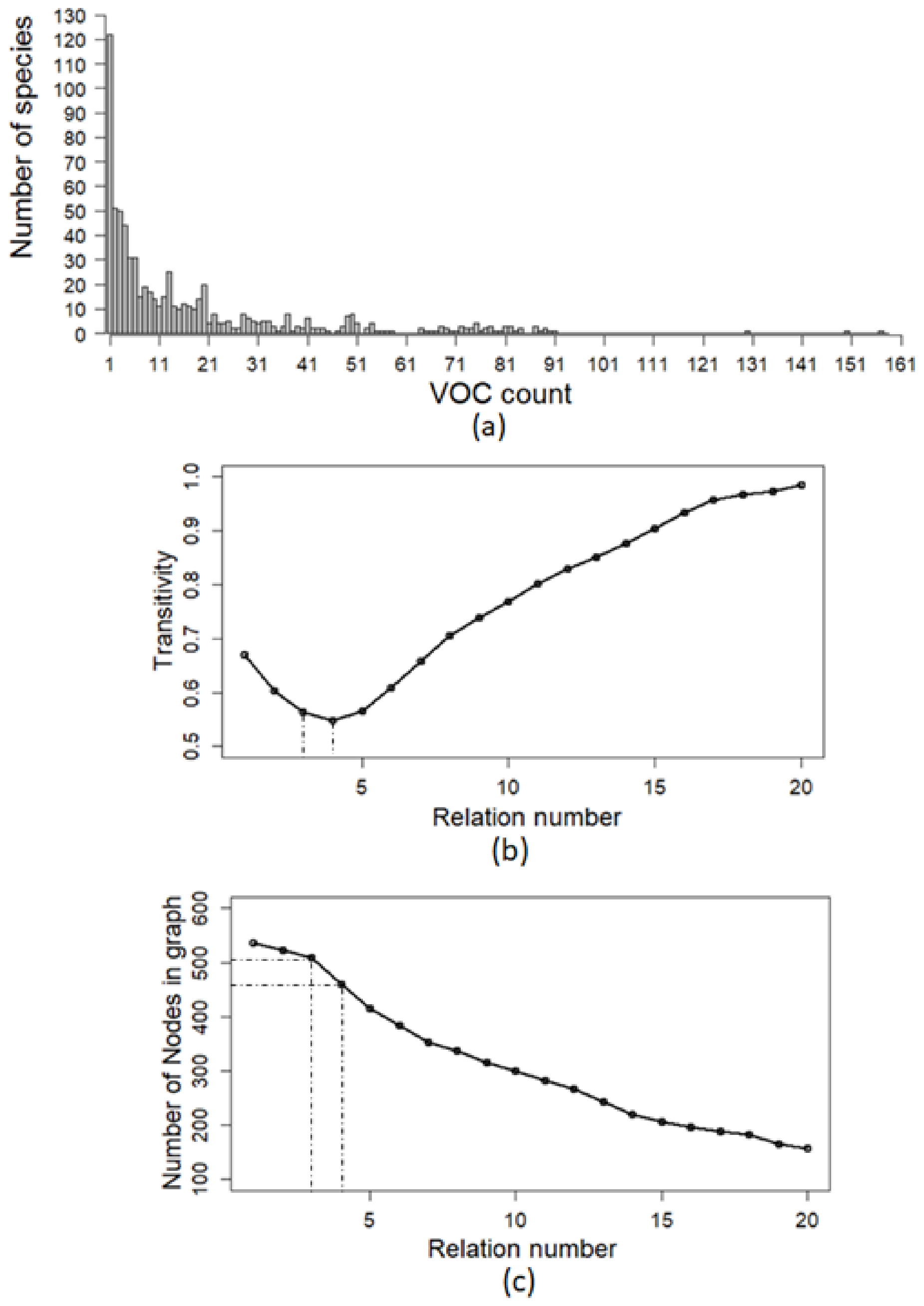


Figure 4. 2 a) Frequency of species vs VOC count, b) Transitivity vs relation number c) Node count vs relation number.

Based on our previous experience we set these DPClusO related thresholds as $CD = 0.5$, $CP = 0.5$ and $OV = 0.05$ [75][76]. Two other important thresholds for the BiClusO algorithm are Th_{rl} and Th_{tf} , which are used to transform the bipartite graph to a simple graph. These thresholds are utilized for the two-step screening of the edges of the simple graph. In this work, we converted the species-VOC bipartite graph to a simple graph where species are the nodes.

A relation number is the number of common VOCs between two species. Therefore, a lower value of Th_{rl} allows many edges in the simple graph which do not represent strong relations. Similarly, we can calculate Tanimoto similarity between any two species in the context of the associated VOCs, and a lower Th_{tf} will allow many noisy edges in the simple graph. At the same time, higher values of Th_{rl} and Th_{tf} will make many species as isolated nodes in the graph, and thus exclude them from the analysis. Therefore, we need to handle the trade-off between allowing meaningful edges and keeping a substantial number of species as non-isolated nodes in the simple graph.

In this work, we determined these thresholds based on the characteristics of the input data, i.e., the speciesVOC relational data. The first step of screening was done based on Th_{rl}. A simple graph can be generated using different values of Th_{rl}. For example, Fig. 4.2(b) shows the plot of Th_{rl} vs. the clustering co-efficient, and Fig. 4.2(c) shows the plot of Th_{rl} vs. the non-isolated species, in the context of the generated simple graphs. In Fig. 4.2(b), initially the clustering coefficient decreases mainly because of the removal of non-important edges. Also, from Fig. 4.2(c), we can see that $Th_{rl} = 3$ allows about 70 more species in the network compared to $Th_{rl} = 4$. In this work, we empirically selected $Th_{rl} = 3$. We then performed the second step of screening based on Th_{tf} . We assume that the relationship between two species is strong if at least 50% of the VOCs of a species match those of the other species. For such a relationship, Th_{tf} should be greater than 0.33 as we prove in the following theorem.

Theorem 1: The Tanimoto coefficient between two binary (0,1) vectors is greater than or equal 0.33 if the common number of 1s between them is greater than or equal 50% of the 1s of any of them.

Proof:

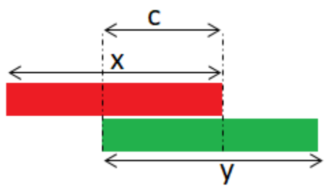


Figure 4. 3 Two overlapping binary vector.

Let the number of 1s in two binary vectors are x and y where c is the common number of 1s between them (Fig 4.3). If c has to be more than or equal to 50% of each of x and y then

we can write $c \geq x/2$ and $c \geq y/2.$

From these conditions, we can write

$4c \geq x + y$

or

$3c \geq x + y - c$

Which is equivalent to $\frac{c}{x+y-c} \geq \frac{c}{3c}$

so,the Tanimoto coefficient $\geq 1/3$

Adding some margin to 0.33, we empirically selected $Th_{tf} \geq 0.4$ for this study. These empirical thresholds produced good results, as explained in the next section.

After observing different test data, we come to the conclusion that a minimum and reasonable value to start with the Th_{tf} is 0.33. As a simple way, the Th_{rl} can be decided by observing the sparseness/density of the input bipartite graph For higher density graphs higher Th_{rl} is recommended. For example, the density of the species-VOC bipartite graph we utilized in the present work is roughly 1% and we used the Th_{rl} as 3. Usually, most practical graphs are sparse (density less than

5%) and using Th_{rl} between 2 to 5 is recommended.

4.4 Results and discussions

In this section, we discuss the results produced by applying the proposed biclustering algorithm to the species-VOC relational data.

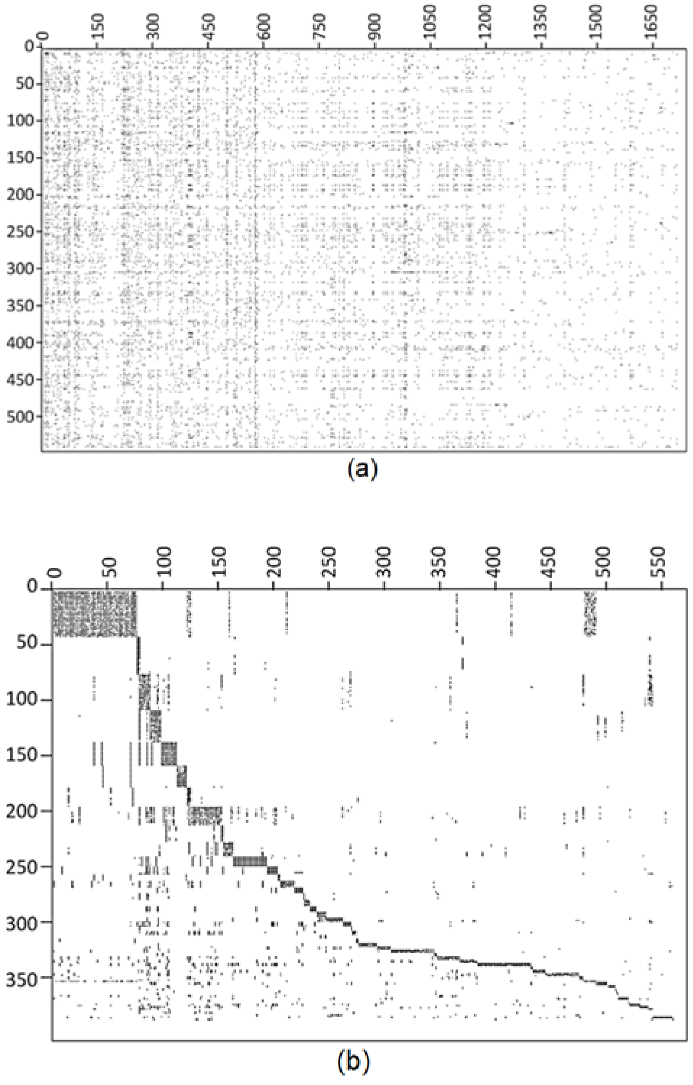


Figure 4. 4 Graphical presentation of matrix data (a) before and (b) after clustering. x axis represents VOCs and y axis represents species.

We examine the properties of the clusters identified by this process, similarities between some of the clusters and some of the VOC groups, and the VOC diversity in terms of kingdoms of species identified by these results. After applying the proposed biclustering algorithm to the species-VOC relational data in this study, we obtained a total of 57 clusters. Fig. 4.4(a) shows the matrix representing the species-VOC relational data before clustering, while Fig. 4.4(b) shows the same matrix after clustering. Visual comparison of Fig. 4.4(a) and 3(b) indicates that our algorithm efficiently separated the data into clusters. The dimensions of the two matrices in Fig. 4.4 are not the same because our algorithm filtered out species and VOCs for which there is not enough reported information, which failed to become part of a meaningful bicluster. The VOC set corresponding to each cluster was determined using *Pvth* $\geq$ *0.6*, which can be considered as characteristic VOCs of the corresponding taxonomic group.

4.5 Properties of clusters

In the present case, a bicluster consists of one set of species and one set of VOCs. First, we assessed the richness of the species of similar taxonomic groups in each cluster.

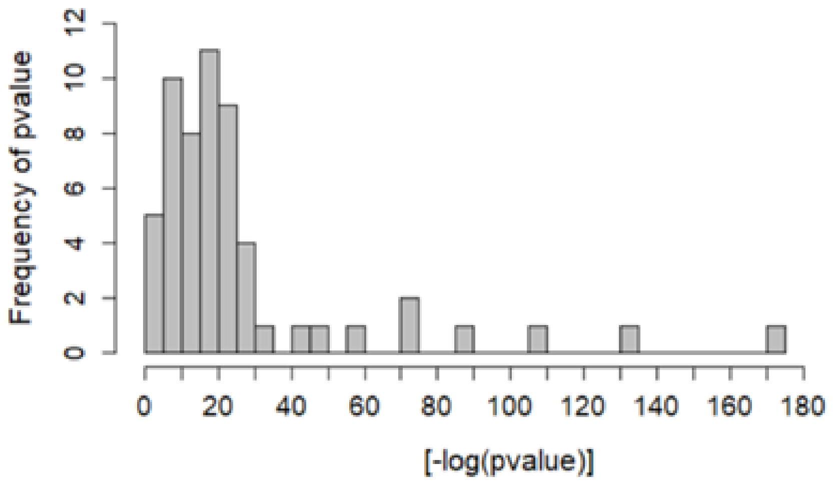


Figure 4. 5 Frequency of p-value of the clusters.

In each cluster, the richness of species belonging to similar hierarchical level, mostly at the family level, was determined by Fisher's exact test. In Supplementary table 1, we summarized all the clusters, referred to as *C1* to *C57*, with their classification, *p-value*, species, and VOCs determined with *Pvth* $\geq$ 0.6. Out of 57 clusters, 36 clusters have *p-values* between 3.19E-52 and 9.46E-05, 12 clusters have values between 0.000134 and 0.006641, and 6 clusters have values between 0.019159 and 0.040969. Only 3 clusters have *p-values* > 0.05. Fig. 4.5 shows the distribution of *p-values*. The low *p-values* for 54 out of 57 clusters are significant, which indicates that the taxonomic classification is consistent with the VOC-based classification.

In the following, we briefly discuss the top 10 clusters based on the *p-value* and the most common VOCs corresponding to the respective clusters obtained by setting *Pvth* = 1 in most cases, i.e., based on bicliques.

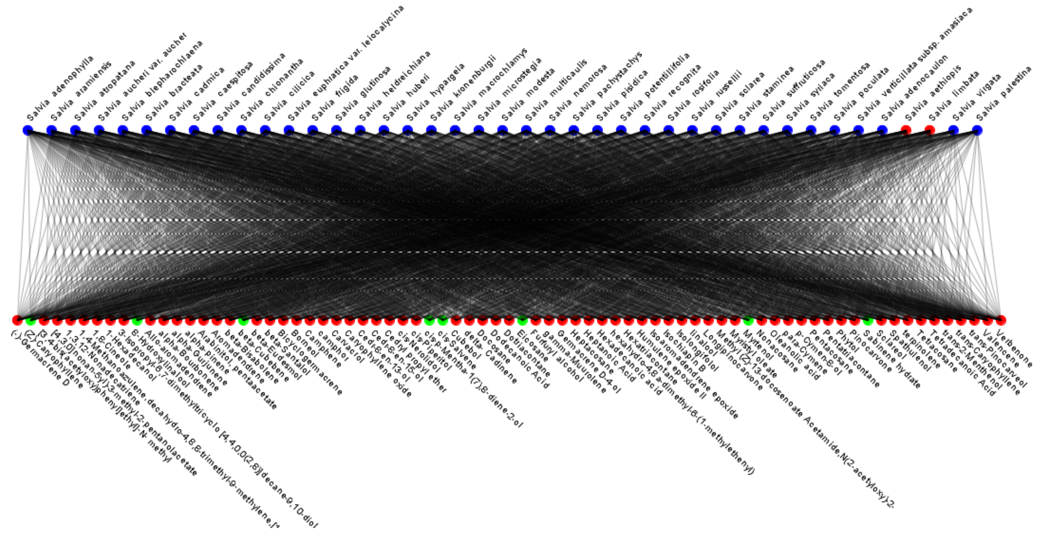


Figure 4. 6 Cluster 1 of lamiaceae family.

In *C1*, there are 45 species of genus salvia under the lamiaceae family (*p-value* 3.19E-52). The most common volatile compounds for this cluster are cedr-8-en-15-ol; humulene epoxideII; p-cymen-8-ol; and *trans*-caryophyllene which form a biclique.

The essential oil composition of the salvia species contains medicinal and aromatic compounds. These species are commonly referred to as members of the mint family.

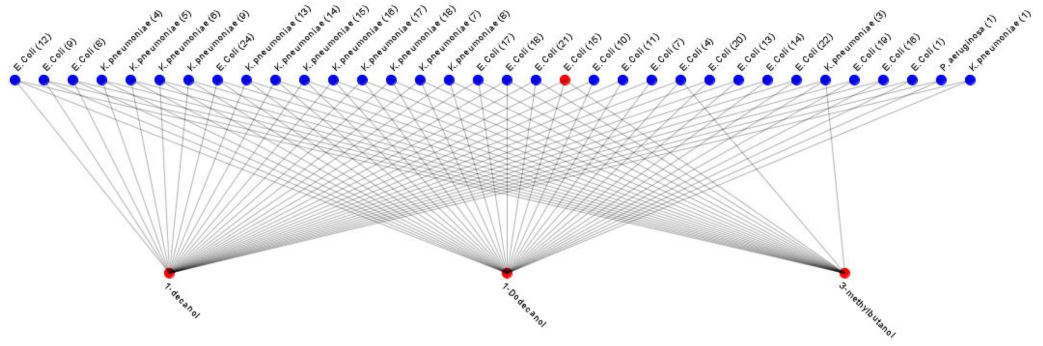


Figure 4. 7 Cluster 2 of enterobacteriaceae family.

C2 is comprised of mostly bacteria in the family enterobacteriaceae (*p-value* 2.38E-32), with the most common VOCs being 1-decanol and 1-dodecanol. All of these species are gram negative bacteria mostly live in animal intestine.

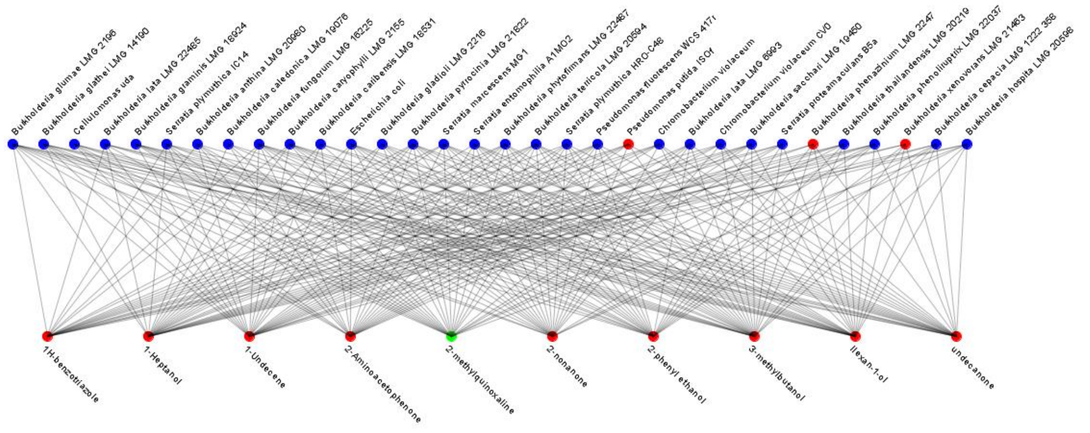

Bukholderia glumae LMG 2196
Bukholderia glathei LMG 14190
Cellulomonas uda
Bukholderia lata LMG 22485
Bukholderia graminis LMG 18924
Serratia plymuthica IC14
Bukholderia anthina LMG 20980
Bukholderia caledonica LMG 19076
Bukholderia fungorum LMG 16225
Bukholderia caryophylli LMG 2155
Bukholderia caribensis LMG 18531
Escherichia coli
Bukholderia gladioli LMG 2216
Bukholderia pyrrocinia LMG 21822
Serratia marcescens MG1
Serratia entomophilia A1MO2
Bukholderia phytofirmans LMG 22487
Bukholderia terricola LMG 20594
Serratia plymuthica HRO-C48
Pseudomonas fluorescens WCS 417r
Pseudomonas putida ISOf
Chromobacterium violaceum
Bukholderia lata LMG 6993
Chromobacterium violaceum CV0
Bukholderia sacchari LMG 19450
Serratia proteamaculans B5a
Bukholderia phenazinium LMG 2247
Bukholderia thailandensis LMG 20219
Bukholderia phenoliruptrix LMG 22037
Bukholderia xenovorans LMG 21463
Bukholderia cepacia LMG 1222 358
Bukholderia hospita LMG 20598
1H-benzothiazole
1-Heptanol
1-Undecene
2-Aminoacetophenone
2-methylquinoxaline
2-nonanone
2-phenyl ethanol
3-methylbutanol
Hexan-1-ol
undecanone

Figure 4. 8 Cluster 3 of burkholderiaceae family.

For *C3*, we could not find any bi-cliques, but setting the coefficient $Pvth \geq 0.8$, we identified four common VOCs, 1-heptanol; 1-undecene; 3-methylbutan-1-ol; and hexan-1-ol. *C3* consists mostly of bacteria from the burkholderiaceae family (*p-value* 5.87E-23). Species in this cluster are mostly pathogenic for humans or animals.

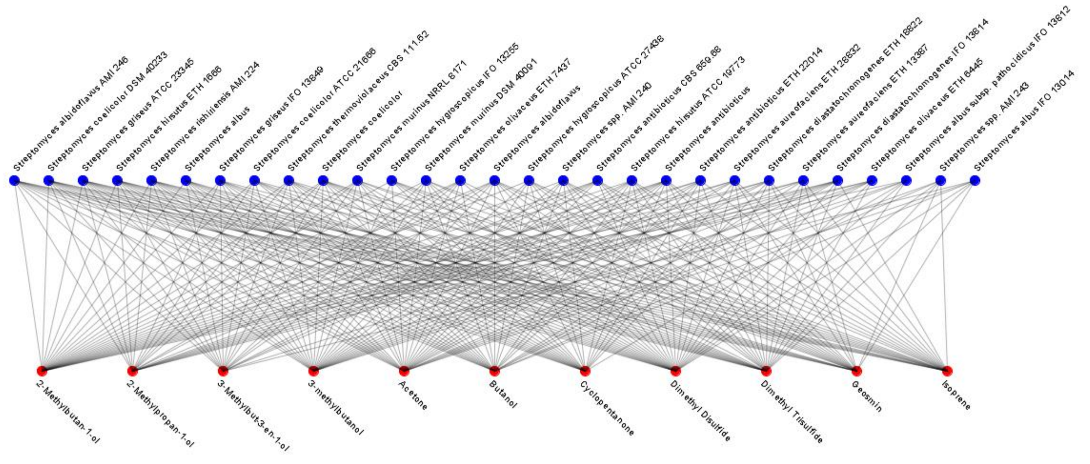

Streptomyces albidoflavus AMI 246
Streptomyces coelicolor DSM 40233
Streptomyces griseus ATCC 23345
Streptomyces hirsutus ETH 1666
Streptomyces ishiriensis AMI 224
Streptomyces albus
Streptomyces griseus IFO 13849
Streptomyces coelicolor ATCC 21666
Streptomyces thermoviolaceus CBS 111.62
Streptomyces coelicolor
Streptomyces murinus NRRL 8171
Streptomyces hygroscopicus IFO 13255
Streptomyces murinus DSM 40091
Streptomyces olivaceus ETH 7437
Streptomyces albidoflavus
Streptomyces hygroscopicus ATCC 27438
Streptomyces spp. AMI 240
Streptomyces antibioticus CBS 659.68
Streptomyces hirsutus ATCC 19773
Streptomyces antibioticus
Streptomyces antibioticus ETH 22014
Streptomyces aureofaciens ETH 28832
Streptomyces diastatochromogenes ETH 18822
Streptomyces aureofaciens ETH 13387
Streptomyces diastatochromogenes IFO 13814
Streptomyces olivaceus ETH 6445
Streptomyces albus subsp. pathocidicus IFO 13812
Streptomyces spp. AMI 243
Streptomyces albus IFO 13014
2-Methylbutan-1-ol
2-Methylpropan-1-ol
3-Methylbut-3-en-1-ol
3-methylbutanol
Acetone
Butanol
Cyclopentanone
Dimethyl Disulfide
Dimethyl Trisulfide
Geosmin
Isoprene

Figure 4. 9 Cluster 4 of streptomycetaceae family.

C4 (*p-value* 1.16E-40) are mostly streptomycetaceae family bacteria which are commercially used to produce antibiotics, antibacterial, antifungal, and antiparasitic metabolites by their secondary metabolism. The common VOC for this cluster is dimethyl disulfide.

C5 (*p-value* 3.87E-23) are mostly bacteria of the leuconostocaceae family. The most common VOCs are 2-methylpropan-1-ol; 2-phenylethanol; 2-phenylpropan-2-ol; 3-(Methylsulfanyl)-1-propanol; 3-methylbutan-2-ol; 3- methylbutanol; benzaldehyde; benzyl alcohol; butanoic acid; decanoic acid; dodecanoic acid; heptanoic acid; nhexaneic acid; octanoic acid; tetradecanoic acid; and valeric acid. These gram positive bacteria can ferment glucose in the heterofermentative way to produce lactic acid.

C6 (*p-value* 1.65E-18) are mostly bacteria of the family prevotellaceae. Most of these

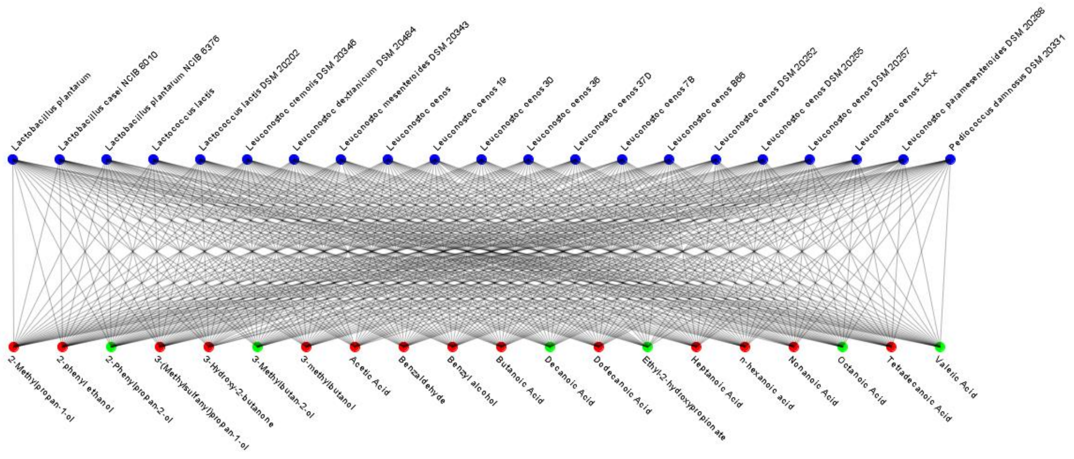


Figure 4. 10 Cluster 5 of leuconostocaceae family.

bacteria are indigenous to the human and animal gastrointestinal tract and oral cavity. The most common VOCs are 12-methyltetradecanoic acid; 13-methyltetradecanoic acid; 3-hydroxy-15- methylhexadecanoic acid; 3-hydroxyhexadecanoic acid; hexadecanoic acid; and tetradecanoic acid.

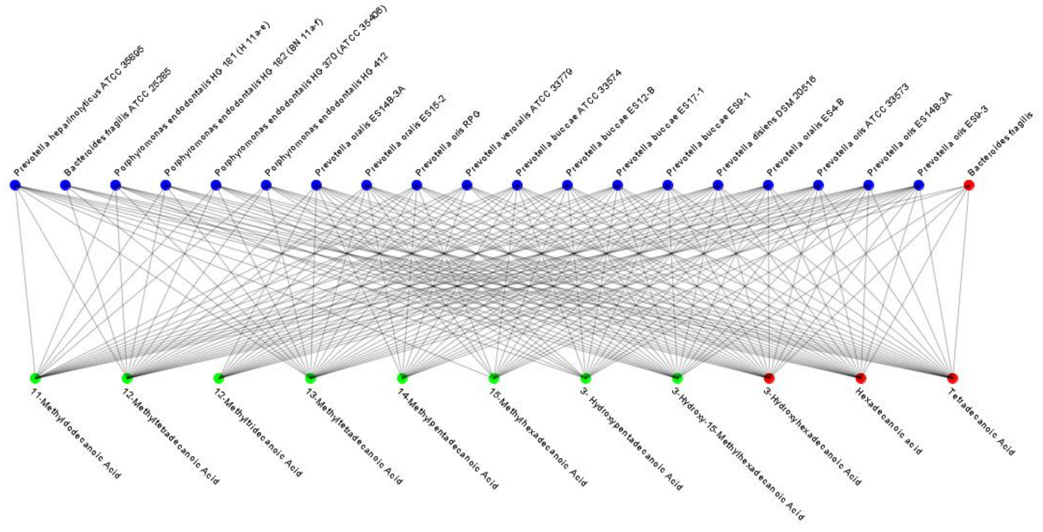

Prevotella heparinolyticus ATCC 35895
Bacteroides fragilis ATCC 25285
Porphyromonas endodontalis HG 181 (H 11a-e)
Porphyromonas endodontalis HG 182 (BN 11a-f)
Porphyromonas endodontalis HG 370 (ATCC 35406)
Porphyromonas endodontalis HG 412
Prevotella oralis ES14B-3A
Prevotella oralis ES15-2
Prevotella oris RPG
Prevotella veroralis ATCC 33779
Prevotella buccae ATCC 33574
Prevotella buccae ES12-B
Prevotella buccae ES17-1
Prevotella buccae ES9-1
Prevotella disiens DSM 20516
Prevotella oralis ES4-B
Prevotella oris ATCC 33573
Prevotella oris ES14B-3A
Prevotella oris ES9-3
Bacteroides fragilis
11-Methyldodecanoic Acid
12-Methyltetradecanoic Acid
12-Methyltridecanoic Acid
13-Methyltetradecanoic Acid
14-Methylpentadecanoic Acid
15-Methylhexadecanoic Acid
3- Hydroxypentadecanoic Acid
3-Hydroxy-15-Methylhexadecanoic Acid
3-Hydroxyhexadecanoic Acid
Hexadecanoic acid
Tetradecanoic Acid

Figure 4. 11 Cluster 6 of prevotellaceae family.

C7 (*p-value* 1.05E-15) are mostly plants of the cannabaceae family. All of these plants are members of the eudicots class, and the most common VOC is betacaryophyllene.

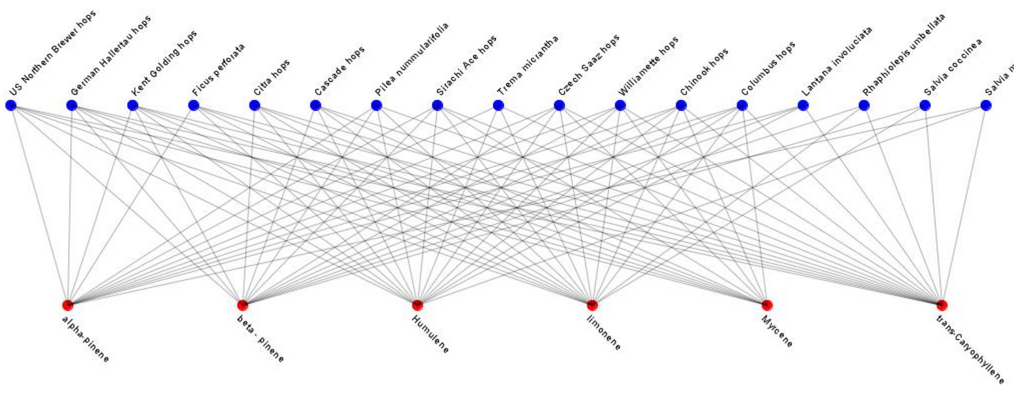


Figure 4. 12 Cluster 7 of cannabaceae family.

C8 (*p-value* 1.92E-27) are mostly fungi of the hypocreaceae family. These species are characterized as opportunistic avirulent plant symbionts. The most common VOCs are 3-methylbutanal; decanal; ethyl acetate; and nonanal.

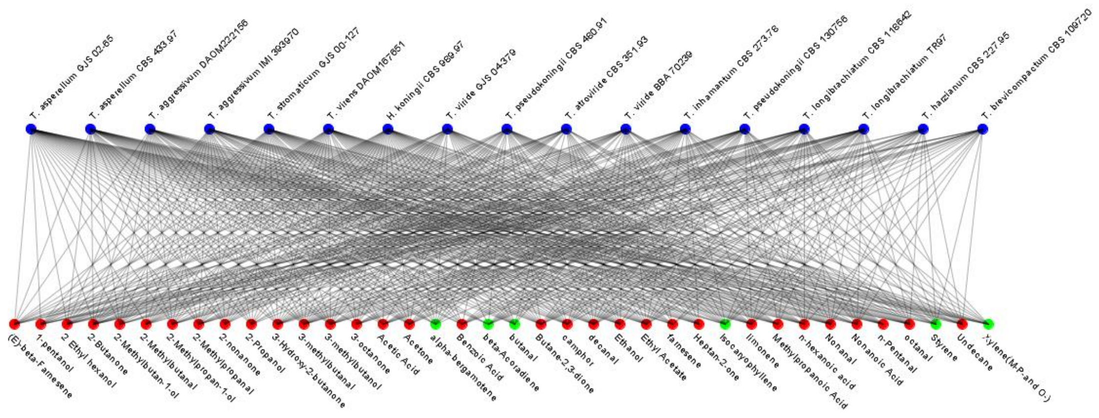


Figure 4. 13 Cluster 8 of hypocreaceae family.

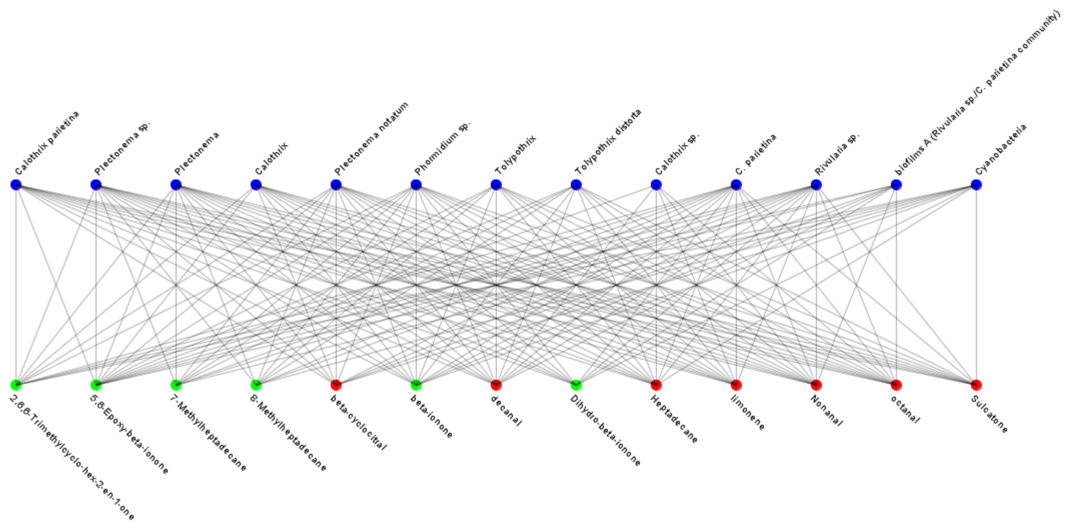


Figure 4. 14 Cluster 10 of cyanobacteria.

C10 (*p-value* 3.82E-10) are mostly bacteria in the cyanobacteria phylum with the common characteristic of obtaining energy by photosynthesis. The most common VOC is beta-ionone-5,6-epoxide.

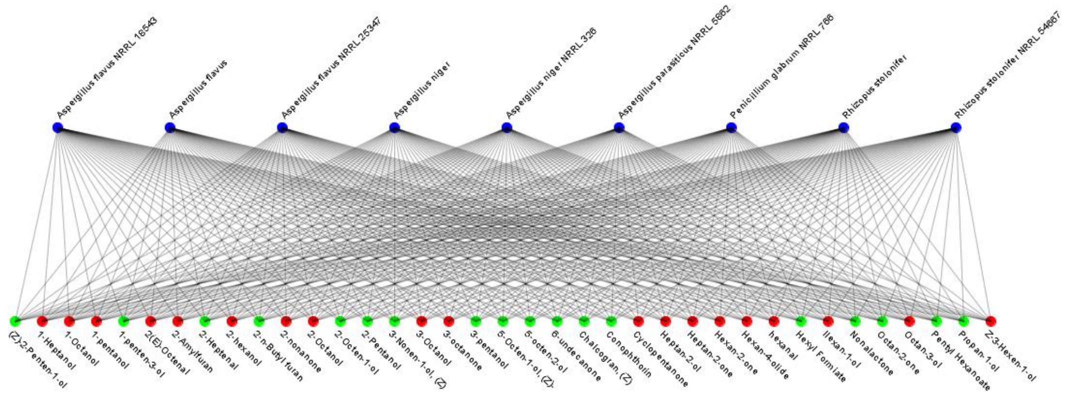


Figure 4. 15 Cluster 11 of trichocomaceae family.

C11 (*p-value* 1.47E-13) are mostly fungi of the trichocomaceae family. These fungi are found in soil and cause disease in corpses. The most common VOCs are (*Z*)-2-penten1-ol; 1-heptanol; 1-octanol; 1-pentanol; 1-penten-3-ol; 2(*E*)- octenal; 2-amylfuran; 2-heptenal; 2-hexanol; 2-n-butylfuran; 2-nonanone; 2-octanol; 2-octen-1-ol ;2-pentanol; 3-nonen-1-ol,(*Z*); 3-octanol; 3-octanone; 3-pentanol; 5-octen-1-ol,(*Z*)-; 5-octen-2-ol; 6-undecanone; chalcogran-(*Z*); conophthorin; cyclopentanone; heptan-2-ol; heptan-2-one; hexan-2-one; hexan-4-olide; hexanal; hexylformate; hexan-1-ol; nonalactone; octan-2-one; octan-3-ol; pentylhexanoate; propan-1-ol; and (*Z*)-3-hexen-1-ol.

4.6 Relationship between clusters

We also assessed the similarity between clusters in terms of shared VOCs. Let a and b be the VOC sets of two clusters, then the percentage of $a \cap b$ in the context of $a \cup b$ is a measure of the Common VOC-based Similarity (CVSim) between those two clusters. In Fig. 4.16, we show the graphs where the nodes indicate the clusters, and the edges indicate a certain minimum CVSim between two clusters. In this figure, the size of a node is proportional to the number of species in it. In Fig. 4.16(a), with minimum CVSim of 10%, we find many edges, implying that many cluster pairs have common VOCs. But, as we increase the CVSim threshold, the number of edges gradually decreases. At CVSim >= 40%, there are only three edges linking 3 pair of clusters: *{C1, C38}*, *{C9, C40}*, and *{C42, C53}*. Cluster *C1* with 41 species and cluster *C38* with three species belong to the plant kingdom of family lamiaceae. Cluster *C42* with three species and cluster *C53* with three species belong to the family enterobacteriaceae in the kingdom of bacteria. Finally, cluster *C9* with 15 species and cluster *C40* with two species belong to the bacteria kingdom in the families bacteroidaceae and veillonellaceae. With CVSim more than 35%, the cluster *C3* with 32 species and cluster *C37* with three species belong to the bacteria kingdom of family burkholderiaceae. Also, in examining these pairs of clusters, we noticed that $|VOC_{C38}| = 74$ and 61 of those are included in VOC_{C1}, $|VOC_{C42}| = 3$ and 2 of them are included in VOC_{C53}, and $|VOC_{C37}| = 8$ and 5 of them are included in VOC_{C3}.

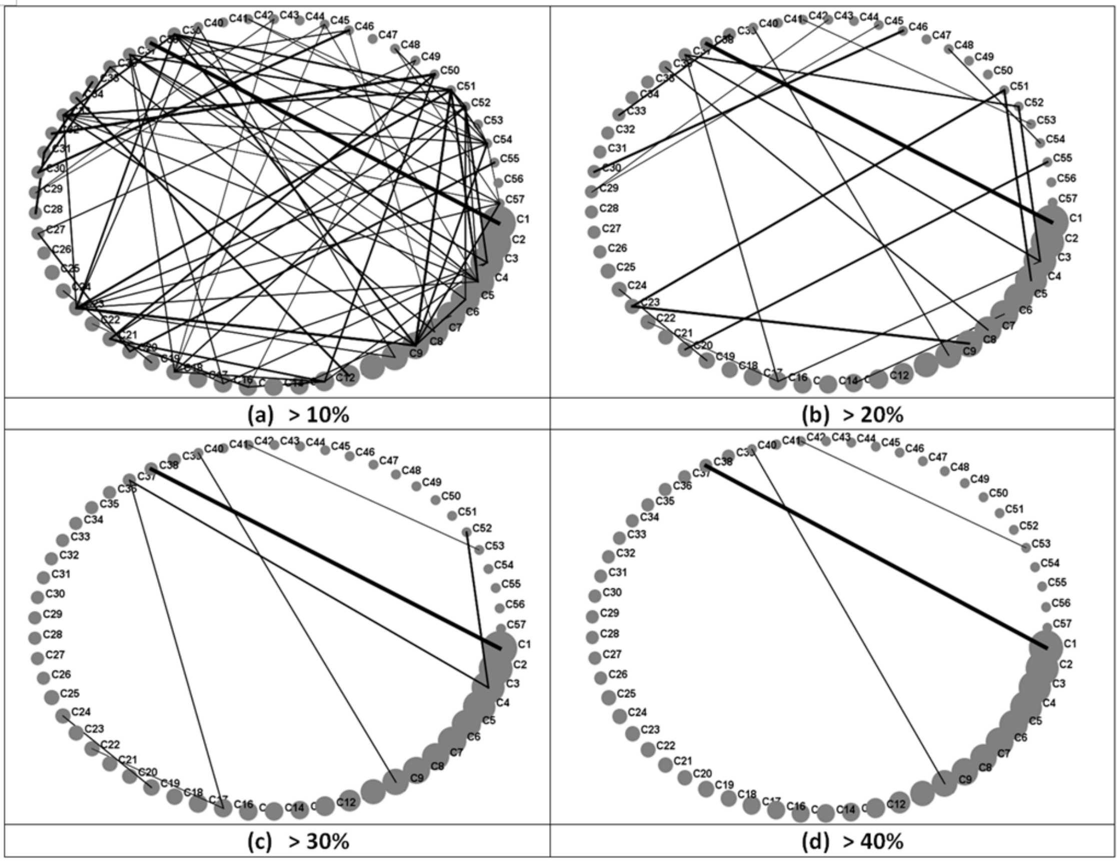


Figure 4. 16 Relations among species clusters in terms of common VOC sets.

Each of these three pairs should be merged into a single cluster because the species belong to the same taxonomical group and the VOCs of one cluster are subsets of another cluster. However, they are divided into 2 clusters mainly because of the lack of enough reported data. Based on such results, we can predict some new species-VOC relations. For example, we can assume that VOCs reported for species of *C1* are likely to be reported for species of *C38* and vice versa, and similarly for the other pairs of clusters.

4.7 Structurally similar VOC groups (SSVGs)

In this section, we focus on relations between clusters of taxonomically similar species with Structurally Similar VOC Groups (SSVGs). This is important because it can be assumed that structurally similar metabolites may belong to the same or related metabolic pathways.

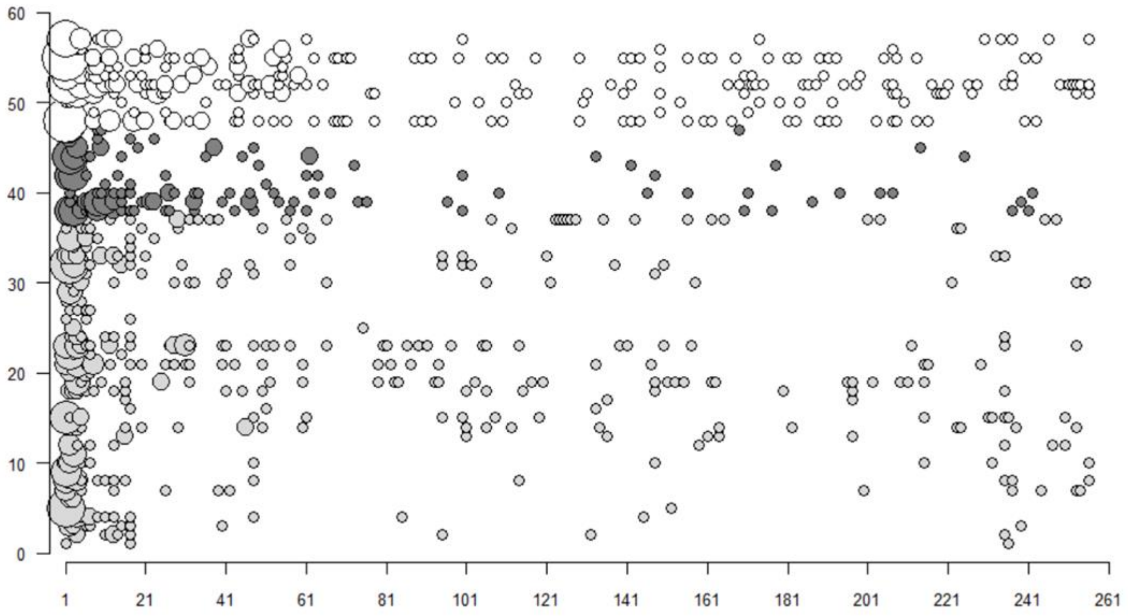


Figure 4. 17 Scatter plot of SSVGs for plant (White), fungi(Black) and bacteria (Gray). x axis represents SSVGs and y axis represents species clusters.

Even though all metabolic pathways in living cells have not been determined, scientists are trying to discover pathways by matching metabolites in terms of their chemical and physical properties [77][78]. VOC biosynthesis depends on transcriptional regulation by which different genes get involved in VOC emissions. Also, post transcriptional regulation may play important roles which yet to be discovered. As the largest producers of VOCs, plants play important roles in the natural ecosystem. In plants, most of the VOCs are generated by using four major pathways: mevalonic acid (MVA), methylerythritol phosphate (MEP), Shikimate or phenylalanine, and lipoxygenase (LOX) [43], [79]. A metabolic pathway is generally defined as some of the consecutive steps of biochemical reactions, catalyzed by enzymes that occur inside a cell.

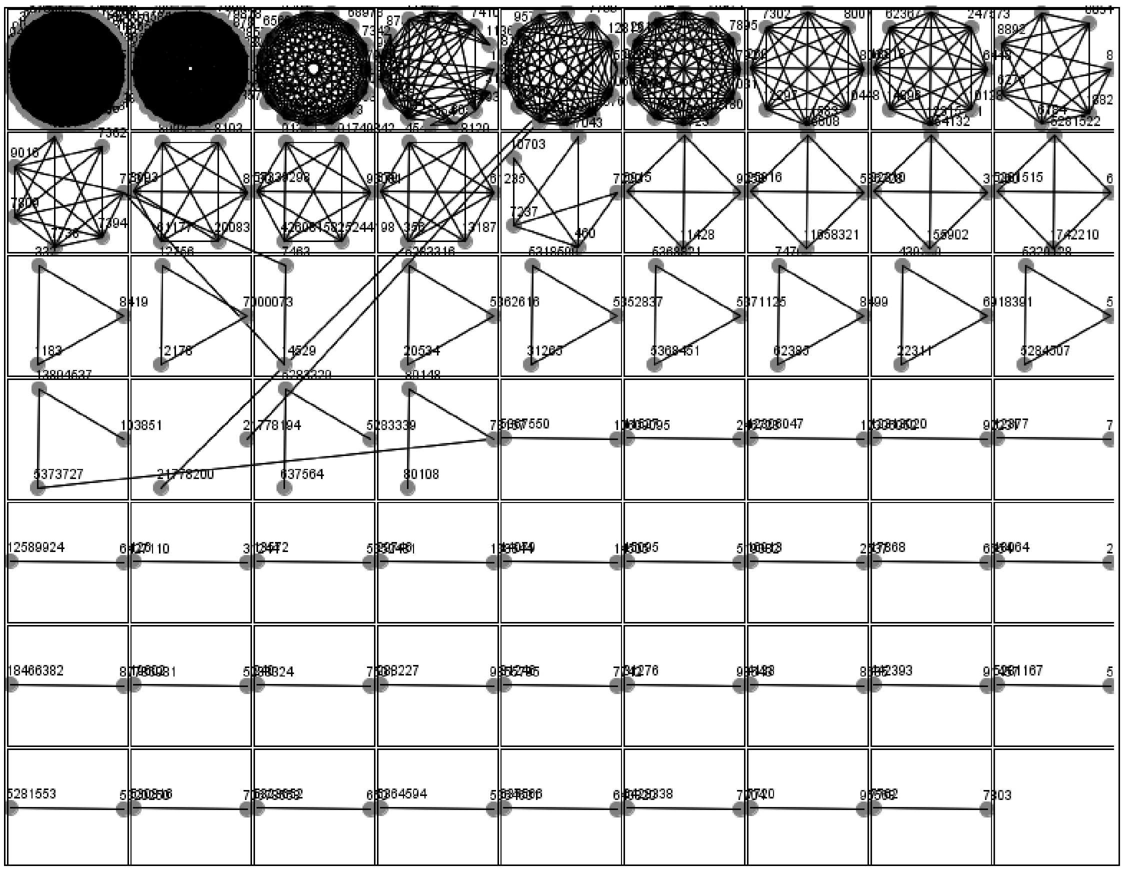


Figure 4. 18 Structurally Similar VOC Groups (SSVGs) with size more than two.

In our study, there were a total of 505 VOCs included in the 57 biclusters generated from the species-VOC relational data. We generated PubChem IDs from the names of the VOCs and then downloaded SDF files for those VOCs and converted them to atom pair fingerprints (APFP) by ChemmineR [80] using the functions "sdf2ap" and "desc2fp" with default parameters. They compute the top 1024 out of 4096 most common atom pairs observed in the compound collection from DrugBank. To determine the structurally similar VOC pairs, we applied the "Tversky" similarity function to those APFP with *cut off* > 0.85, *alpha* = 1 and *beta* = 1. These coefficient

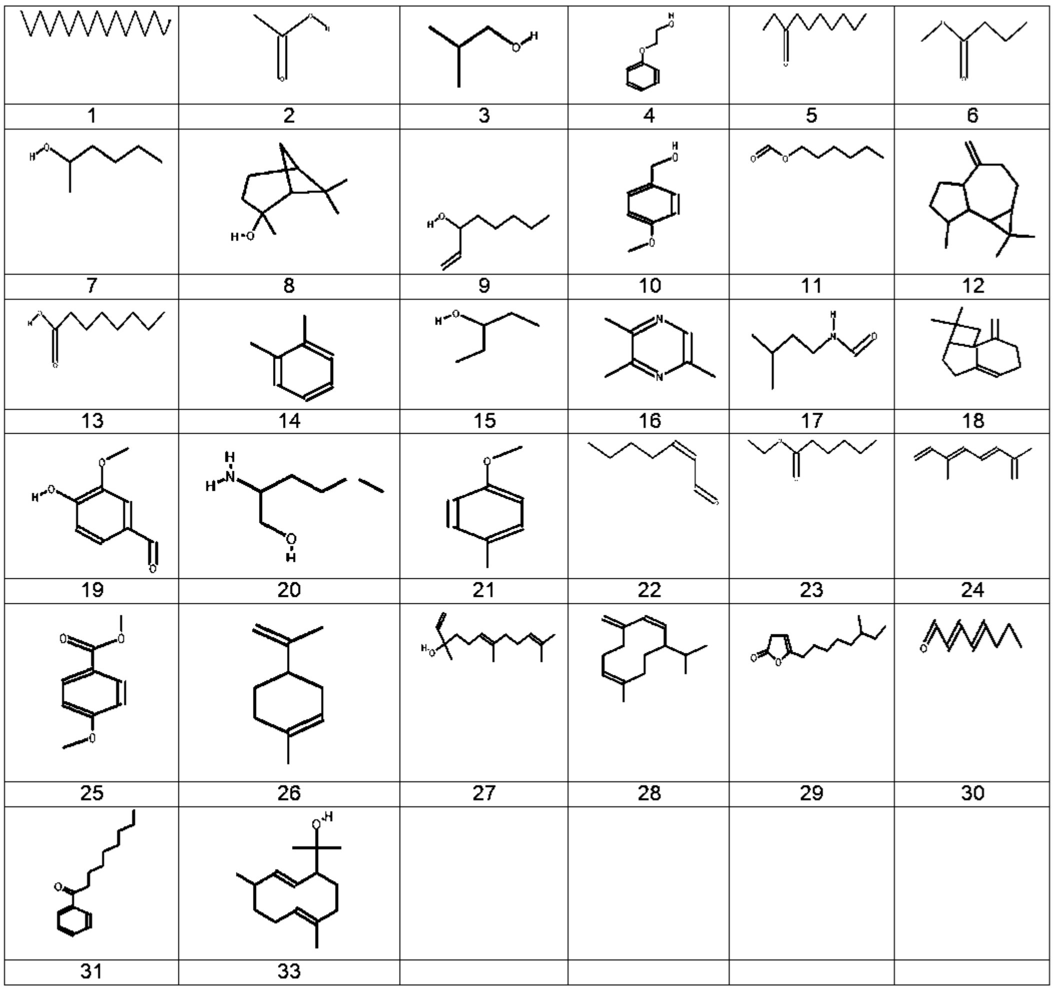


Figure 4. 19 Chemical 2d structure of a representative element in an SSVG (For SSVGs with size at least 3).

values are recommended as the best to measure similarity among compounds [81], [82]. Thus we constructed a network of VOCs based on the chemical structure similarities between them. Then we applied the DPClusO tools [75] to create clusters from this network with *CD* = 0.7, *CP* = 0.5 and *OV* = 0.05. Doing this clustered the total 505 VOCs into 256 SSVGs, of which 62 are of size 2 or more and the rest are single VOCs. Fig. 4.18 shows the cluster result (SSVGs) with at least two VOCs. Fig. 4.19 shows the chemical 2d structure of a candidate element in each of the SSVGs with size of 3 or more, where we can visually see that the structures corresponding to different SSVGs are different. Supplementary figure 1 shows the structure of all elements of the first four clusters.

Fig. 4.17 shows the mapping of the VOCs belonging to 57 clusters to 256 SSVGs. When one or more VOCs belonging to a cluster matched the VOCs of a SSVG, we placed a circle at the corresponding location on the map. The size of a circle on the map in Fig. 4.17 is proportional to the number of shared VOCs between a cluster and a SSVG.

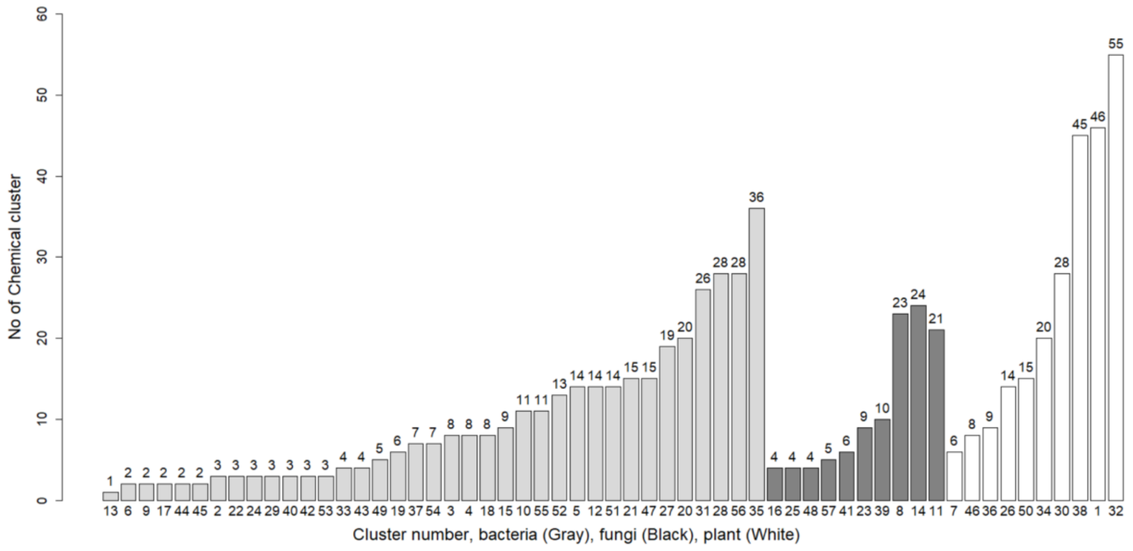


Figure 4. 20 Number of associated SSVGs in different species groups.

The color of a circle is white, black, or gray depending on whether the corresponding cluster belongs to the kingdom of plants, fungi, or bacteria. From the mapping, cluster *C1* is linked to 46 SSVGs of which the first 9 groups are SSVG1, SSVG8, SSVG11, SSVG18, SSVG21, SSVG27, SSVG26, SSVG28 and SSVG32. In Fig. 4.19, the representative compounds for these 9 groups are displayed with labels 1, 8, 11, 18, 21, 27, 26, 28, and 32. The sparsity of the map in Fig. 4.17 implies that the species of different taxonomical groups produce different types of VOCs in terms of chemical structure.

It can be hypothesized that structurally different VOCs are produced by different metabolic pathways. Thus, it may be suggested that the characteristic VOC pathways in different taxonomical groups are different. Such pathways evolved depending on the needs for survival and adaptation in the environment for certain classes of organisms. Further investigation into species-specific VOCs can provide more clues to the interaction of the species with the ecosystem.

4.8 VOC diversity across three kingdoms

Using the first taxonomical hierarchy of species, the kingdom, we classified the biclusters into three groups, i.e., plants, fungi, and bacteria. First, we compared plants, fungi, and bacteria in terms of the abundance of associated characteristic SSVGs. Out of 57 clusters, 10 clusters belong to plants, 10 belong to fungi, and 37 belong to bacteria. The histogram in Fig. 4.20 shows the number of associated SSVGs with different clusters in the three kingdoms.

To determine the differences between these three kingdoms in terms of the number of associated SSVGs to individual clusters, we performed a *t-test*. Based on the *t-test*, the *p-values* are: plants-fungi, 0.04934; plants-bacteria, 0.0295; and bacteria-fungi, 0.06914. From the results of the *t-test*, we can say that the groups of plants have more diversified VOCs compared to individual groups of bacteria or fungi. This is consistent with the fact that plants are a more advanced species compared to bacteria and fungi, and thus require more diversified VOCs for survival. Also, plants have different organs such as roots, leaves, stems, flowers, and fruits which can produce different amounts and types of VOCs at different times, e.g., different climate seasons, flowering seasons, diurnal and nocturnal times, for different purposes. Also, we conducted a *t-test* with *p-value* 0.04028 between eukaryotes and prokaryotes, i.e., bacteria vs. combined clusters related to plant and fungi. The *p-value* indicates that individual groups of eukaryotes produce more diversified VOCs than individual groups of prokaryotes.

We also assessed the diversity of VOCs across the three kingdoms in terms of different chemical structures. Generally, each of the SSVGs can be considered as a group of chemical compounds related to similar or related metabolic pathways. In our dataset, most clusters were generated involving bacteria because we found more data related to bacteria in published literature. Separately, plants are associated with 136 SSVGs, fungi with 58 SSVGs, and bacteria with 148 SSVGs. Although individual bacterial groups are associated with smaller numbers of SSVGs, the total

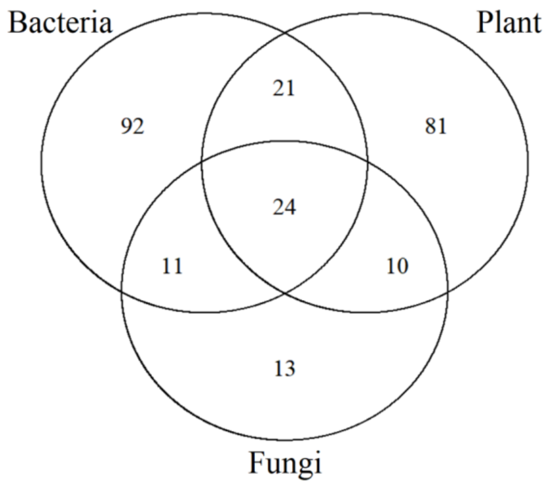


Figure 4. 21 Venn diagram showing the overlap of different SSVGs among three kingdoms.

number of SSVGs related to bacterial clusters is the highest. This is because diversified types of bacteria live under diversified environments requiring diversified types of VOCs. Also, plants are associated with more diversified types of VOCs compared to fungi because plants are a more advanced species than fungi.

The Venn diagram in Fig. 4.21 shows the sharing of SSVGs across the kingdoms of plants, fungi, and bacteria. The evolutionary hierarchy of the tree of life reveals the gradual decline or uptake of biochemical traits by descendant species from their ancestors. The tree of life generated using DNA data from 3000 species [83] reveals that bacteria are a predecessor to both fungi and plants, and that plants are a predecessor to fungi. DNA, converted to mRNA, creates different types of proteins, and proteins engaged in different biochemical reactions eventually generate VOCs. Fig. 4.21 shows that many VOC pathways are conserved in all three kingdoms, or between any two kingdoms. The number of pathways conserved between plants and bacteria are more than those conserved between bacteria and fungi or plants and fungi. Plants and fungi probably acquired different VOC pathways from different bacterial species at different times through different interactions by horizontal gene transfers. Fig. 4.21 shows plants are related to 81 unique SSVGs, fungi to 13, and bacteria to 92. Different bacteria species living in many different types of microenviroments developed many unique types of VOCs. Plants being more advanced (need to handle different biological processes) and sessile (face different challenges of environmental stress) organisms developed many unique VOCs for their survival. Fungi are less advanced organisms and different types of fungi live in very similar environments, so they can survive with a somewhat smaller number of VOCs.

4.9 Conclusion

We applied the BiClusO algorithm to species-VOC relationship data to identify groups of species in terms of common VOCs. Based on Fishers exact test we showed that VOC-based classification of species is consistent with their taxonomical classification. Inter-cluster relationships identified in terms of VOCs can help to predict the VOCs of new species in a similar taxonomical group. Furthermore, along with the algorithm, we used DPClusO and ChemmineR to identify structurally similar VOC groups (SSVGs). The relationships of those SSVGs to different groups of species imply that common VOC pathways in different taxonomical groups are different. Based on the results of the *t-test*, we showed that individual groups of plants are associated to more VOCs compared to groups of bacteria. Further analysis of the relations of these groups of species with SSVGs revealed the diversity of VOCs across the kingdoms of plants, fungi, and bacteria. Pairwise overlapping of SSVGs among three kingdoms reveals the conservation of evolutionary hierarchy in terms of biochemical traits.

Chapter 5

Application of BiClusO to miRNA-mRNA bipartite data

Multidimensional data mining from integrated environment of different data sources is frequently performed in computational system biology. Molecular mechanism from the analysis of complex network of gene-miRNA can aid to diagnosis and treatment of associated disease. In this work we mainly focus on finding inflammatory bowel disease (IBD) associated microRNAs (miRNAs) by bi-clustering of miRNA-target interactions aided by known IBD risk genes and their associated miRNA collected from several sources.

5.1 Introduction

Inflammatory bowel disease (IBD) generates disorders in different digestive organ parts with prolonged pain and disruption. The specific causes of IBD disease which eventually leads to ulcerative colitis (UC) and Crohn's disease (CD) remain unknown.

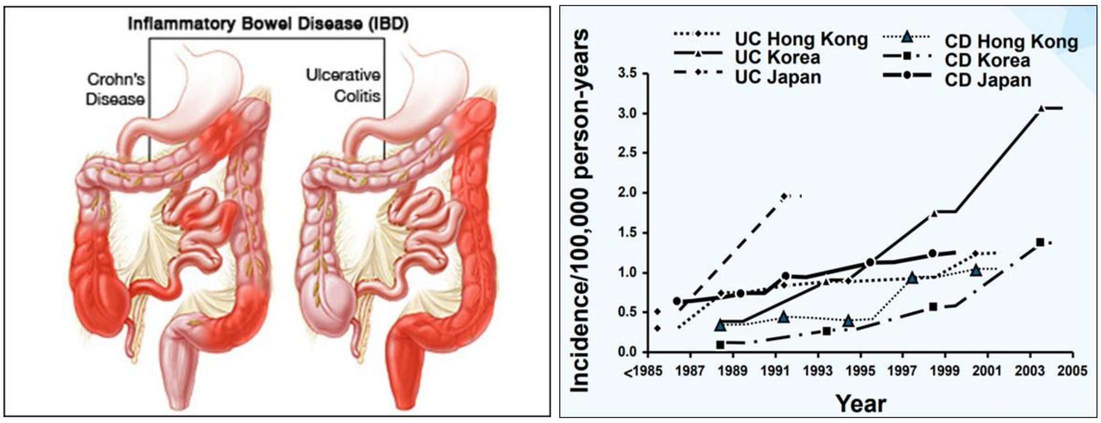


Figure 5.1 Epidemiology of Inflammatory Bowel Disease.

Epidemiology of IBD shows (Fig 5.1 Edward V. Loftus, Jr.), M.Dthe increasing rate by every year where prevention or cure of this disease is still intractable[84]. Even different risk factors, such as ethnicity, smoking, age, family history, gender are attributed to the IBD disease, scientists are trying to find other evidences by analyzing the genomic data related to it. MicroRNAs (miRNAs) are non-coding RNAs of approximated length of 22 nucleotides, playing important roles in gene splicing and post transcriptional regulation of gene(Fig 5.2 (From google)). miRNA functions are as follows: cell cycle control, apoptosis, hematopoiesis, hypoxia,cardiac and skeletal muscle development, neurogenesis,insulin secretion, cholesterol metabolism, aging, immune responses and viral replication. Recent study revealed that there is strong connection between regulatory mechanism of miRNA and disease etiology. Scientists have developed miRNA - target interaction (MTIs) databases based on different proven scientific methods which can be used to drill down the functional modules of specific miRNA sets and their target interactions. Previously we have developed methods to identify the IBD associated genes from integrated analysis of transcriptome data and protein-protein interactions (from HIPPIE database). We also compared our result with three different databases namely HuGENet, DisGeNet, CTD

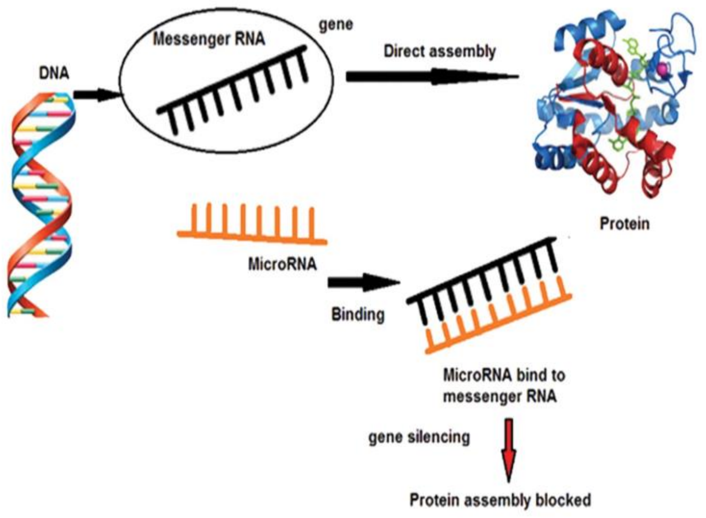


Figure 5. 2 Mechanism of miRNA in cell.

and another genome wide association study (GWAS) with respective IBD genes of 849, 866, 129 and 335. Finally we identified a group of IBD related genes with different confidence score[76]. A miRNA-regulatory module (MRM) is a subset of MTIs where a groups of miRNAs participate cooperatively by regulating a bunch of genes to control different biological processes[85]. In the current work we mainly focused on MRMs detection from MTIs involving the IBD related genes. We searched these genes in different MTIs database by using a new biclustering approach[24]. We evaluated the relevance of the miRNAs with IBD by counting their occurrences in different MRMs and their interactions with known IBD genes. Finally we normalize the score of each miRNA for different MTIs database and evaluate the importance of different miRNA.

5.2 PPI network construction

5.2.1 Curating IBD gene

Based on the IBD gene expression data downloaded from NCBI's Gene Expression Omnibus (GSE57945) [86], we got 1197 and 4315 differentially expressed genes (DEGs) (with false discovery rate (FDR) < 0.05) between control and Crohn's disease (CD) as well as control and ulcerative colitis (UC) samples, respectively.

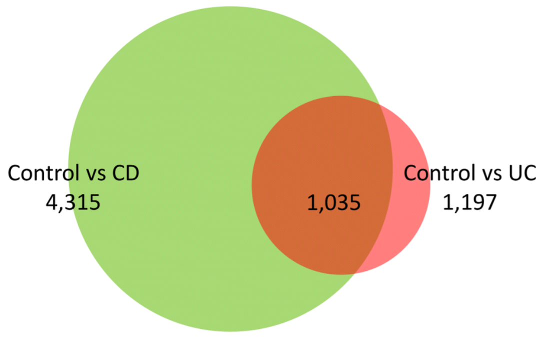


Figure 5. 3 venn diagram.

The venn diagram of the overlapping genes between these two sets is shown in Fig. 5.3. CD and UC are closely related diseases, hence, the differentially expressed genes are largely overlapped (1035 overlapped genes). As our focus is to find novel IBD genes and pathways by system level analysis, we took the union set of the differentially expressed genes from these two comparisons, and combined these genes to a single set consisting of 4477 genes. The differentially expressed genes are the potential candidates to be relevant to IBD.

5.2.1 Construction of a disease relevant PPI network

We initially downloaded 866 genes reported in DisGeNet database [87] as IBD genes. We found that 318 of the 866 IBD genes are out of the 4477 differentially expressed genes (DEGs) we identified from gene expression analysis. Let us name these 318 genes as IBD related differentially expressed genes (IDEGs) and the rest 4159 as only differentially expressed genes (ODEGs). In this work we consider these 318 genes as known IBD genes.

We constructed a disease related PPI network based on Human Integrated Protein-Protein Interaction rEference (HIPPE) database [88]. In HIPPE database each interaction is reported with a confidence score. We first extracted the interactions involving ODEGs with a score greater than 0.7, which included 4135 ODEGs. We then retrieved the interactions involving all 318 IDEGs with a score greater than 0.1. Thus we retrieved a total of 38,500 interactions involving IDEGs, ODEGs and other genes (OGs). From these interactions, we empirically selected interactions to construct the final PPI network according to following criterion: IDEG-IDEG:0.1, IDEG-ODEG:0.1, IDEG-OG:0.72, ODEG-IDEG:0.1, ODEG-ODEG:0.1, ODEG-OG:0.85. In summary, we gave the highest priority to interactions involving IDEGs (genes that are both known IBD genes and differentially expressed genes according to the expression data we used). Also, most priority was given to interactions for which both genes are ODEGs (only differentially expressed genes). These genes are likely to contain system level information of molecular mechanism of IBD. The HIPPIE database recommend 0.72

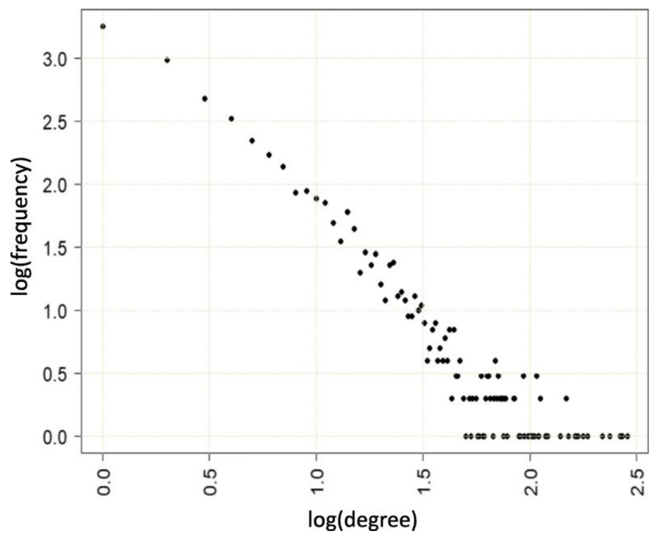


Figure 5. 4 Degree distribution of the IBD related PPI network follows the power law.

as a good score which we used for IDEG-OG interactions and finally adjusted 0.85 for ODEG-OG interactions to roughly keep similar number of DEGs (IDEGs + ODEGs) and OGs (Other Genes) in the PPI network for the sake of balance and thus extracted unbiased information. Finally we selected 16,429 interactions involving 5056 genes with 291 IDEGs, 2072 ODEGs and 2693 OGs. The degree distribution of the network is shown in Fig. 5.4. As many other typical PPI networks, the degree distribution of our constructed network followed power law. Some other global network properties of the network include average path length 4.18, clustering co-efficient 0.1 and diameter 11. For such a big network the clustering coefficient of 0.1 is substantialy enough indicating presence of densely connected clusters in the network.

5.2.2 Clustering of the PPI network

After creating the disease related PPI network we determined clusters in the network by DPClusO algorithm. DPClusO generates overlapping clusters and ensures coverage. For example, each node goes to at least one cluster. We hypothesize that clustering of a disease relevant PPI network helps isolate systems with disease related properties and therefore statistically significant clusters enriched with known IBD genes can be used to predict novel IBD genes and pathways based on the associations determined by combined information of IBD gene expression and protein-protein interactions.

We generated 9 sets of clusters from the PPI network by DPClusO algorithm using density values of 0.1, 0.2, 03, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9. Table 5.1 shows characteristics, i.e. the number of clusters, size of the biggest cluster and average cluster size, related to the clusters generated by the 9 different density values. As expected, smaller density value resulted in larger and fewer number of clusters generated.

Table 5. 1 Characteristices of the clusters generated with different input densties using the DPClusO algorithm based on the IBD related PPI network.

Density	**Noofcluster**	**Maxsize**	**Avgsize**
0.1	827	70	18.52
0.2	1229	42	10.69
0.3	1779	31	7.18
0.4	2219	21	5.74
0.5	2790	16	4.61
0.6	3597	13	3.53
0.7	4425	11	2.59
0.8	4534	9	2.50
0.9	4775	7	2.31

To assess the enrichment of IDEGs in each of the identified clusters we determined Fisher's exact test *p*-values. In this work we proposed to consider statistically significant clusters for predicting novel IBD related genes and pathways. Therefore we assigned a score called SScore (Significance Score) to each gene as a measure of confidence of prediction based on the *p*-values of the clusters they belong to. The definition of SScore is provided in the Methods section. Based on these scores we performed ROC analysis to determine which set of clusters should be used for predicting novel IBD genes.

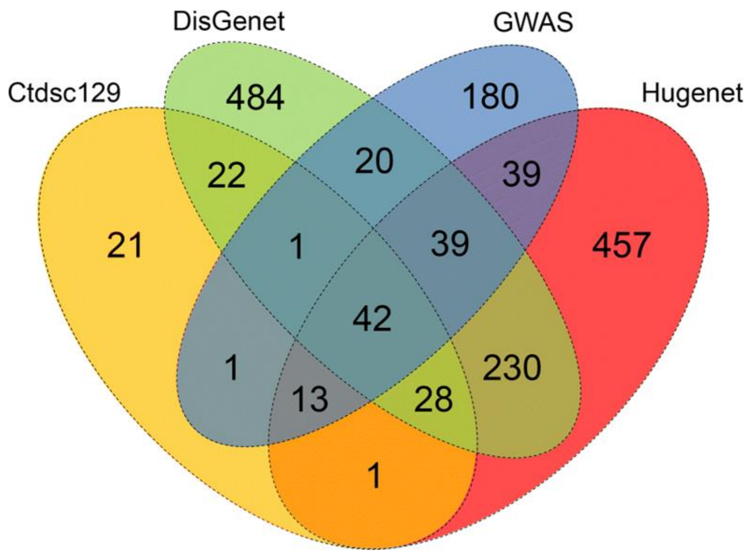


Figure 5. 5 Venn diagram showing overlapping between IBD genes collected from four different sources.

5.2.3 ROC analysis

In our disease relevant PPI network there are total 5056 genes out of which 291 genes are IDEGs which are among the 318 genes considered as known IBD genes in the present work. We predicted the degree of relevance of the rest 4765 genes with IBD based on SScore. We collected well curated and well studied IBD genes from 3 databases as follows, The Comparative Toxicogenomics Database (CTD) [89], DisGeNet [87], HuGENet [90] and published literatures on results of GWAS [91]-[94]. The venn diagram of the reported IBD genes in these 4 databases is shown in Fig. 5.5. It is noticeable that IBD genes listed by these 4 sources are substantially different, indicating the need for finding comprehensive set of potential IBD genes. Although these four sources are not the complete list of IBD genes, they can be used to assess the effectiveness of SScore. The ROC curves corresponding to the 9 sets of clusters are very similar, which imply the underlying signal in the carefully constructed PPI network is very strong and DPClusO algorithm has been successful to catch the signal for across a wide range of the input parameter. Fig 5.6 shows the Area Under the Curve (AUC) for the 9 ROC curves. The AUCs are not very high, which may be due to incomplete information of known good quality IBD genes. We observed that the highest AUC was obtained in the case

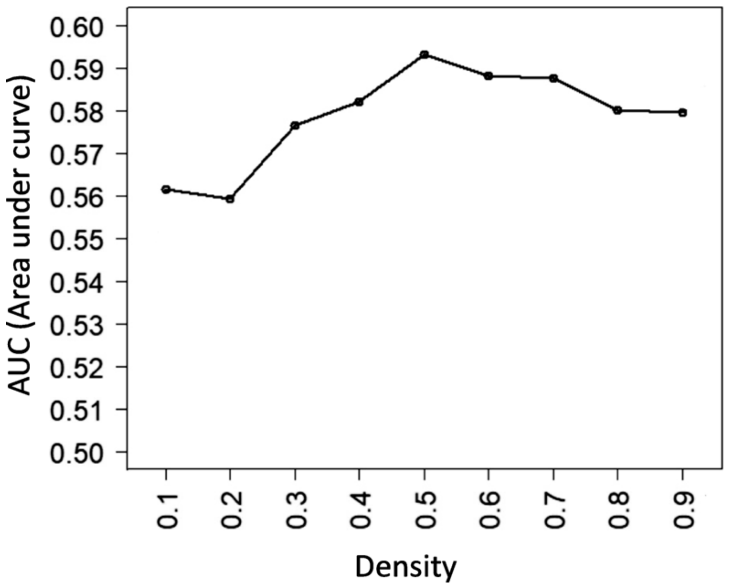


Figure 5. 6 AUCs corresponding to 9 sets of clusters.

of the cluster set generated using density =0.5. So we selected the genes included in the statistically significant clusters of this set, adjusted the corresponding *p*-values for multiple testing [95] and selected the genes having adjusted p-values less than 0.05 as predicted IBD genes.

5.2.4 Prediction and validation

We predicted 909 genes (with adjusted $p-value<0.05$) included in the clusters selected from the set corresponding to the highest AUC as our predicted IBD genes. These 909 genes are other than the genes considered as known IBD genes (IDEGs) in this work. The list of the 909 predicted IBD genes and corresponding adjusted p-values are shown in Additional file 1. To validate our results we initially searched how many of the predicted genes are exactly matched with well curated known IBD genes. We found 83, 8, 54, 22 of the predicted genes matched with reported IBD genes in (1) HuGeNet, (2) CTD, (3) DisGeNet databases and (4) GWAS results respectively. After considering overlapping between databases, 14.5% of our predicted genes matched with good quality known IBD genes. Given the fact that we made predictions based only on a specific gene expression data and a limited set of known IBD genes, the 14.5% matching with good quality data is significant ($p-value<3.45\times10^{-12}$, p-value determined based on hypergeometric ditsribution assuming total number of human genes as 20000). However, our approach is a computational approach. So, it is rational to compare our result also with computationally predicted IBD genes. In CTD database other than the good quality curated set there is a big set of genes inferred as IBD genes by various methods. When we compare our result with this big set, we find that 93.8% of the genes we predicted matched with reported IBD genes ($p-value<9.8\times10^{-14}$). As we have predicted the genes by wisely integrating the information of gene expression and protein-protein interaction, it is very likely that they are truely related to IBD. One of the predicted genes IL12B is supported by all four above-mentioned sources as an IBD related gene. IL12B and IL23R have been identified as susceptibility genes for IBD by recent genome-wide association studies [96]. Each of the three genes CCR5, IL1R2 and LTA is mentioned as IBD related gene in three of the above mentioned sources. High expression of CCR5 has been reported in active IBD [97]. Epithelial

IL1R2 takes part in homeostatic regulation during remission of ulcerative colitis [98]. It has been reported that LTA elicits a strong inflammatory reaction controlled by intestinal dendritic cells [99]. Thus we have found IBD relevance of many other predicted genes by literature review. The proposed method, however is a computational one and the role of the newly predicted genes in IBD pathogenesis should be clarified by further studies.

The degree of relevance of the 909 genes predicted by the proposed approach can be evaluated by the corresponding *p*-values. The top 20 predicted novel IBD genes (not reported in any of the four sources) based on *p*-values are IKBKG, BIRC3, BCL10, RNF31, RBCK1, CCRL1, LAMC3, CARD11, KISS1, THBS2, TRAF2, TRAF1, PYCARD, MIS12, ALB, AR, RIPK1, SHARPIN, SNAPIN and ITGA2B. Many of these 20 top IBD risk genes we identified from this study have been found to be associated with IBD. In human, the IKBKG gene encodes NF- *κ*B essential modulator (NEMO) which is an inhibitor of nuclear factor κB kinase subunit gamma (IKK- γ) [100]. NEMO (IKK- *γ*) is the regulatory subunit of the inhibitor of the I- *κ*B kinase (IKK) complex, that activates NF- *κ*B causing activation of genes involved in inflammation, immunity, cell survival, and other pathways. IBD-like immunopathology can be developed by IKBKG [101]. BIRC2 and BIRC3 are important genes in regulating the expression of proinflammatory cytokines, such as TNF- *α*, through NF- *κ*B and MAPK pathways [102]. BCL10 is an adaptor protein which is assumed to play role in the PAF-induced inflammatory pathway in human intestinal epithelial cells [103]. RNF31 and HOIL-1L complex functions in linear ubiquitination of proteins in the NF- *κ*B pathway in response to proinflammatory cytokines [104]. CCRL1 acts as a functional receptor for the monocyte chemoattractant protein family of chemokines; elevated chemokine expression is associated with many inflammatory diseases such as IBD, rheumatoid arthritis and asthma [105][106]. As a component of the LUBAC complex, RBCK1 conjugates linear (Met1-linked) polyubiquitin chains to substrates and thus plays imoportant role in NF- *κ*B activation and inflammation regulation

[107]. RBCK1-deficiency is associated with autoinflammatory syndrome and immunodeficiency [107]. LAMC3 is expressed saliently at significantly different proportions in low and high coherence expression profiles of IBD patients [108]. The elevated stromal protein thrombospondin-2 (THBS2) has been reported to be a part of a fibroblast-specific inflammation signature [109]. It has been shown that TRAFs are important mediators of innate immune receptor signaling [110]. IBD and IBD recurrence is associated with the overexpression of TRAF2 [111]-[113]. TRFA1 is reported to be highly expressed in IBD patients [114]. To form the basic Inflammasome subunit, the adaptor protein ASC (encoded by the PYCARD gene) links the NLR sensor to caspase-1 [115]. TNF- α-induced necroptosis is associated with two members of the receptor-interacting protein (RIP) family of kinases – RIPK1 and RIPK3 [116]. Tumor necrosis factor- α (TNF- α) can bind to one of two receptors, TNFR1 or TNFR2; TNFR activation results in the activation of NF- κB leading to the induction of proinflammatory cytokines [116].

5.2.5 Comparison with ToppGene

It has been demonstrated that ToppGene [117] performs better than several other methods such as SUSPECTS [118], PROSPECTOR [119], ENDEAVOUR [120] in candidate gene prioritization. From the ToppGene suite [121] we used ToppGenet which is a web based tool that can take input a set of seed genes and can return a list of genes with closely related roles with a prioritization score. In our work, based on gene expression data and DisGeNet database we considered 318 genes as known IBD genes and based on those we predicted 909 other genes as IBD related genes. We assigned the same 318 genes to ToppGenet and from the output we selected the highest ranking 909 genes which we compared with the 909 genes determined by our approach. For both sets, we determined the number of genes matched with the union of reported IBD genes in 4 sources. Also we determined the AUCs using prioritization score and SScore in case of ToppGenet and our approach respectively.

In case of ToppGenet, we selected network based approach as our approach is also network based. Furthermore, we used 3 available options for ToppGenet as follows: (i) K-Step Markov, (ii) Page rank with priors and (iii) Hits with priors. The comparison results are shown in Table 5.2. The results show that performance of our approach is comparable in terms of the number of identified genes and better in terms of AUC.

Table 5. 2 Results of comparison with ToppGene.

Parameter of comparison	ToppGenet			Our Method
	K-step Markov	Page rank with priors	Hits with priors	
Number of match	163 >	169 >	102 <	132
AUC	0.4969 <	0.4339 <	0.4831 <	0.5826

5.3 Unify the IBD gene set

HuGENet, DisGeNet, CTD and another genome wide association study (GWAS) with respective IBD genes of 849, 866, 129 and 335. Our method discovered total 909 genes which has p-value <= 0.05. We unified these newly discovered genes with HIPPIE, HuGENet, DisGeNet, CTD and another genome wide association study (GWAS). Total 2245 genes were found. These 2245 genes are used to search the associated miRNA from different database.

5.4 Finding micro RNA regulatory module

5.4.1 miRNA-mRNA Interaction dataset

We selected four different database from online which are mirWalk

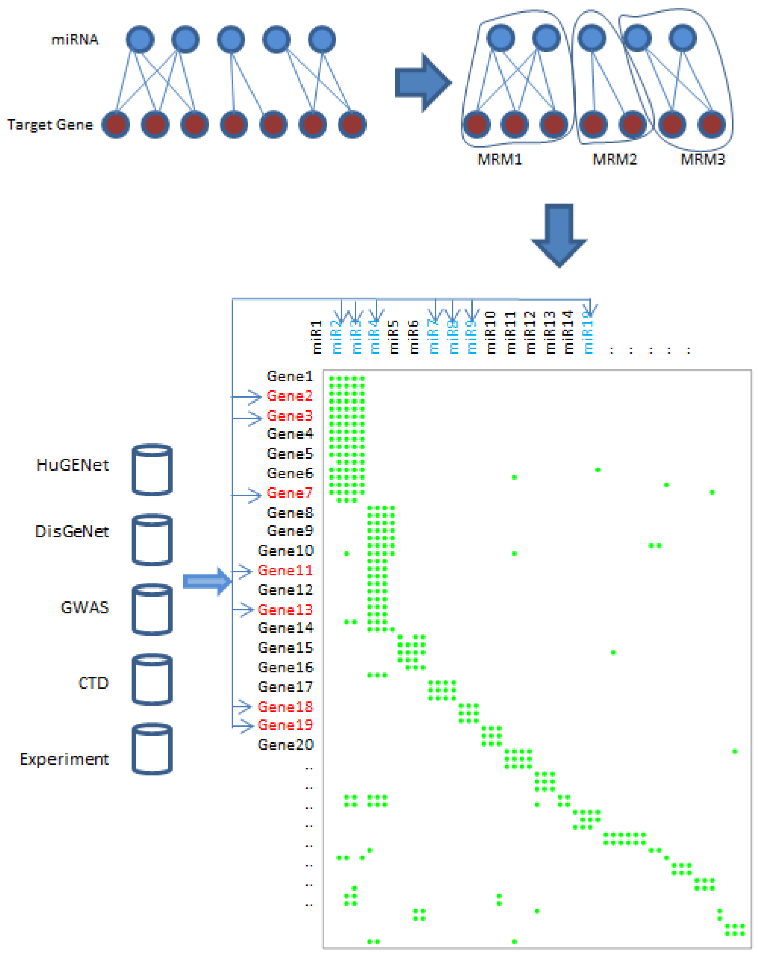


Figure 5. 7 Flow of the proposed approach; Finding MRMs (upper). Mapping IBD genes in MRMs and finding corresponding miRNAs (lower).

(http://mirwalk.umm.uni-heidelberg.de/)[122], DIANA (http://diana.imis.athena-innovation.gr)[123], miRecords (http://c1.accurascience.com)[124], miRTarbase (http://mirtarbase.mbc.nctu.edu.tw)[125]. We used at least four rows counting for each interaction of DIANA dataset which confirms the minimum two tissue samples on each interaction. On mirWalk dataset, we considered at least three different descriptions of each interaction. We applied "Support Type" filtering using the string value 'Functional MTI' on the miRTarbase dataset. Filtering extracts the significant relationship from the dataset which forms a sparse matrix. The following Table 5.3 shows the information of different dataset after applying filter.

Table 5. 3 Four dataset with number of interactions miRNA and Gene.

Dataset	**Interactions**	**miRNA**	**Gene**
mirWalk	17290	51	4621
miRTarbase	8157	735	2756
miRecords	1710	249	1097
DIANA	17535	521	5491

5.4.2 Sub MRM extraction:

A MTI network forms a bi-partite graph of miRNA and mRNA where modules can be extracted by bi-clustering which is similar to biclique enumeration approaches. Fig 5.7 shows the method of extracting MRMs from a MTI network. At first a Biclustering method was applied to the MTI network of a dataset which generates a bicluster set. We used BiClusO biclustering algorithm which was developed in our lab shows best performance over some renowned biclustering algorithm[24][126]. BiClusO algorithm generates a precise number of overlapping bicluster algorithm under optimized parameter settings[24][76]. We took cluster density=0.5, cluster property=0.5, relation number=3, Tanimoto coefficient =0.33 and attachment probability =0.5. Each bicluster is called an MRM. A typical MRM is constructed by strongly connected miRNAs and genes. A subset from an MRM is the desired gene set denotes the possible set of genes with specific disease or functions. The different categorical or functional dependency of such desired gene set can be extracted from these MRMs.

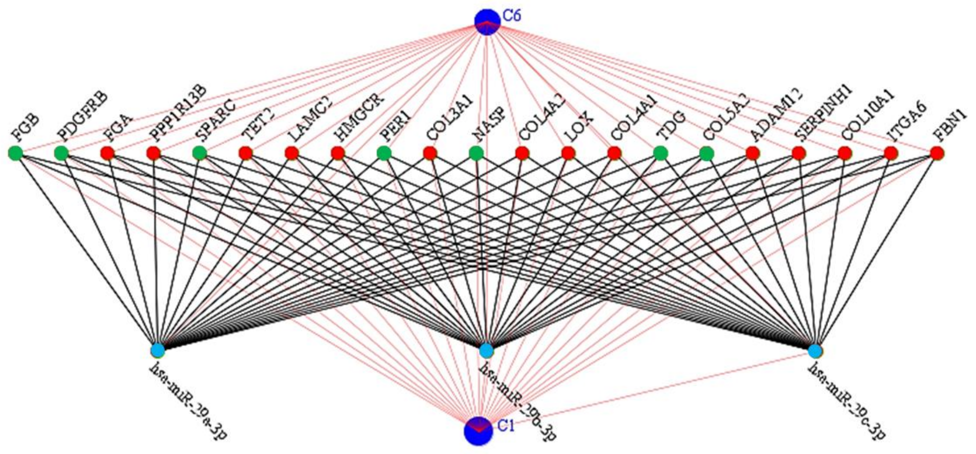


Figure 5. 8 A typical sub-MRM from an MRM.

In our study, the 2245 genes are possible gene set of IBD disease. For each bicluster, these genes were matched and corresponding miRNAs were separated. Thus submodule of individual MRM was generated. Fig 5.8 shows a typical sub-MRM from a MRM. The green colored nodes in the gene side and corresponding miRNAs forms a sub-MRM from miRTarbase dataset. Almost every case, the total number of sub-MRM was less than the number of MRMs and size of each sub-MRM was less than the corresponding size of the MRM.

5.4.3 Score calculation

Total number of interactions, miRNAs and genes are different for each datasets in our experiment. Even we downloaded the latest updated version the collecting method and importance of each interaction are varied in terms of different parameters for different data source. Combined score calculation emphasis on relevance score and existence of each miRNA on more than two dataset (Fig 5.9). Usually there is no dispute over a miRNA which has been cited on each datasets and should have a high rank. After generating the sub-MRMs we calculated the relevance score of individual miRNA by using the following formula

RSmiRNA(i) = NoofIBDmiRNA(i) * NoofclustermiRNA(i)

Where RSmiRNA(i)= Relevance score of ith miRNA

NoofIBDmiRNA(i) = number of IBD genes attached to ith miRNA in IBD MRM set

NoofclustermiRNA(i)= number of IBD MRMs attached to ith miRNA

The relevance score signifies the attachment of each miRNA to different of sub-MRM and a subset of IBD genes. The total number of interactions, miRNAs, and genes are different for each dataset in our experiment.

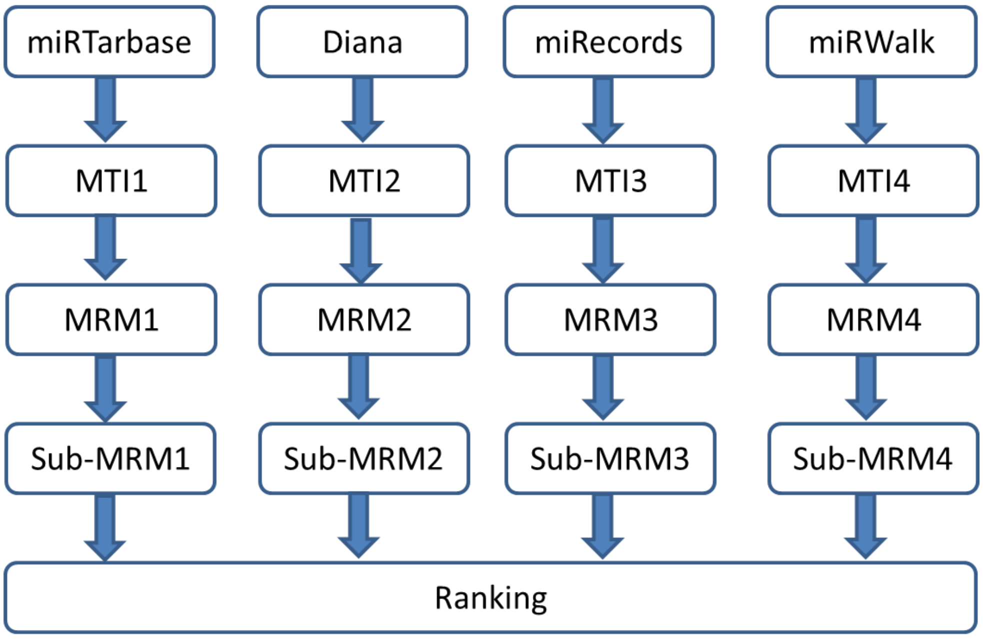


Figure 5. 9 Combined score using different database.

After finding the miRNA sets and their corresponding relevance score from four different datasets all sets were unified. We normalized the score of individual miRNA in each dataset by using the following formula

$$\boldsymbol{TSmiRNA_i = \sum_{n=1}^{4} \frac{RSn_i}{Cn_i} \sum_{n=1}^{4} En_i} \qquad 5.1$$

Here

$TSmiRNA_i$ is the total score of ith miRNA all dataset

RSn_i is the relevance score of ith miRNA in nth dataset

Cn is the number of cluster in nth dataset

En_i is the Boolean value of ith miRNA found in nth dataset

As an example, hsa-let-7b-5p was found in three datasets DIANA, mirTarbase, and mirWalk. The total number of biclusters generated from DIANA, mirTarbase, and mirWalk datasets are 650, 64 and 1579. In DIANA hsa-let-7b-5p was attached to 44 biclusters with 48 IBD genes. In mirTarbase it was attached with 1 bicluster with 1 IBD gene and in mirWalk, it was attached with 209 biclusters with 69 IBD genes. So the relevance score for this miRNA in three datasets are 2112, 1, and 14421. The total score is (2112/650+1/64+14421/1579)(1+1+1) = 37.21

5.5 Results

5.5.1 Ranking miRNA

miRWalk dataset generates 1579 biclusters where we found 1011 sub-MRMs with 50 miRNAs and 333 genes. Top 10 miRNAs by relevance scores are hsa-let-7d-5p, hsa-let-7a-5p, hsa-let-7e-5p, hsa-let-7c-5p, hsa-let-7b-5p, hsa-miR-106a-5p, hsa-miR-106b-5p, hsa-let-7f-5p, hsa-let-7i-5p. 64 biclusters with 41 sub-MRMs encompassing 100 miRNAs and 128 genes were found in mirTarbase dataset. Top 10 miRNAs are hsa-miR-221-3p, hsa-miR-29b-3p, hsa-miR-222-3p, hsa-miR-34c-5p, hsa-miR-200c-3p, hsa-miR-29c-3p, hsa-miR-200b-3p, hsa-miR-29a-3p, hsa-miR-34b-3p, hsa-miR-24-3p. 23 biclusters were generated by

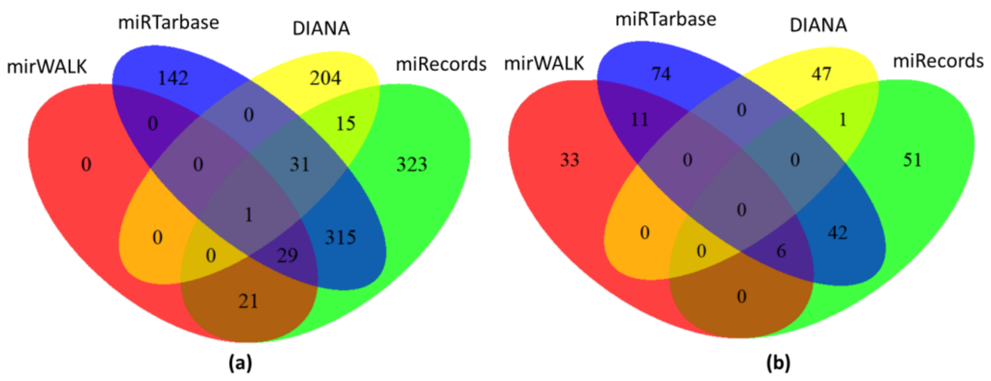


Figure 5. 10 Number of miRNAs in different dataset a) before biclustering b) after biclustering.

small dataset miRecords where 20 sub-MRMs with 48 miRNAs and 54 genes were found. Top 10 miRNAs are hsa-miR-16, hsa-miR-15a, hsa-miR-17, hsa-miR-29a, hsa-miR-181a, hsa-miR-29b, hsa-miR-1, hsa-miR-221, hsa-miR-20a, hsa-miR-34b.

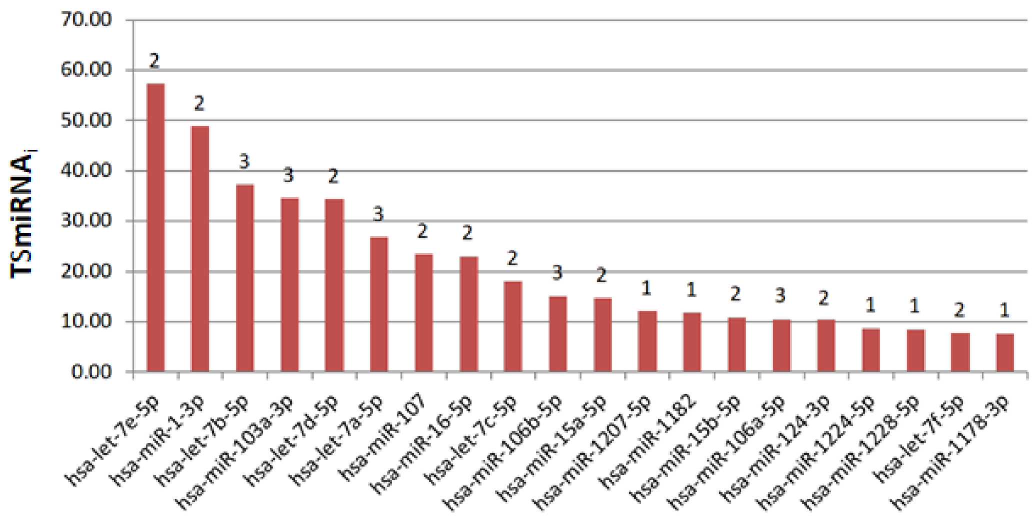


Figure 5. 11 Total score of top 20 miRNAs with number of attachment to different datasets.

DIANA dataset generates 650 biclusters with 423 sub-MRMs where 133 miRNAs and 340 genes were found. Top 10 miRNAs in this dataset are hsa-miR-1-3p, hsa-miR-16-5p, hsa-miR-15a-5p, hsa-miR-15b-5p, hsa-miR-124-3p, hsa-miR-103a-3p, hsa-miR-27a-3p, hsa-miR-107, hsa-miR-20a-5p, hsa-let-7b-5p. Venn diagram from Fig 5.10 shows the number of miRNAs in different datasets before biclustering (a) and after biclustering and detection of sub-MRMs (b). From the Venn diagram, it is evident that most of the datasets have an almost distinct set of miRNAs. Scoring formula extracts 5 common miRNAs found in three different datasets and 50 common miRNAs on two different datasets. Fig 5.11 shows the top 20 miRNA according to the total score where 15 of them were found in at least two datasets. Supplementary table 2 shows the list of miRNAs with repective total relevance score.

5.5.2 Relevance of IBD associated disease

In our top 20 miRNAs, a 6 miRNAs are related to the miR-7 family where most of them have a high score. miR-7 family has 9 members which are let-7a, let-7b, let-7c, let-7d, let-7e, let-7f, let-7g, let-7i, and miR-98. microRNAs related to this family regulate various biological functions such as cell proliferation, cell cycle, stem cell biology, metabolism, and migration, progression, and chemoresistance. miR-7 is downregulated on different types of cancer where Colon cancer[127], gastric tumors[128] are found. Patients with inflammatory bowel disease (IBD) are at significantly increased risk of colorectal cancer (CRC)[129][131], principally resulting from the pro-neoplastic effects of chronic intestinal inflammation.[132]

Using imperfect base pairing to the 3'-UTR the mature let-7 negatively regulates the expression of target mRNAs at a posttranslational level[133]. The expression levels of let-7 microRNAs in stem and progenitor cells are maintained low during the normal development process. Expression levels increase when the progenitor cells differentiate [134] The downregulation of let-7 promotes migration and invasion of normal intestinal epithelial cells and CRC cells[135]. The downregulation of let-7 or upregulation of either LIN28A or LIN28B has been reported to be related to the prognosis in CRC patients in critical stage. The expression level of LIN28B was inversely correlated to that of mature let-7a in human CRC [135]. From an experiment, 38 % out of 600 CRC patients were found to be highly expressed of LIN28A or LIN28B [136]. Let-7 microRNAs are also downregulated in different types of cancers such as hepatocellular carcinoma (HCC), gastric adenocarcinoma, pancreatic cancer, ovarian cancer, prostate cancer, Burkitt lymphoma, renal cell carcinoma, breast cancer, and melanoma [137]. Expression of hsa-let-7e-5p is markedly upregulated in HHM RC. Subsequent assessment of the expression of hsa-let-7e-5p

target genes implicated that it may be a prognostic biomarker for RC with HHM[138]. Both inflamed and non-inflamed terminal ileal mucosa in adult patients with active CD have their distinct miRNA expression patterns compared with healthy controls for hsa-let-7b-5p[139]. let-7d has a significant impact on epithelial-to-mesenchymal transition (EMT) and formation of cancer initiating cells which are resistant to irradiation and chemical exposure and responsible for cancer metastasis[140]. In patients with stage II CRC hsa-miR-103a-3p is reported as a promising predictive biomarkers for tumor recurrence [141].Expression of miR-16 is elevated in CD and UC peripheral blood [142]. Overexpression of miR-106b-5p suppress the CRC cell migration and inhibits the invasion and metastasis of colorectal cancer by targeting CTSA [143]. Upregulation of miR-15a-5p in IBD patient is reported in [144]. By sponging miR-1207-5p a long noncoding RNA BC032469 upregulates hTERT expression which promotes proliferation in gastric cancer[145]. There was a significant negative correlation between miR-1182 and hTERT which attenuates gastric cancer. miR-15b-5p is down-regulated in CRC cells and tissues. The inhibitory effects of miR-15b-5p on cell apoptosis and enhancement of drug sensitivity are mediated by the down-regulation of its NF-κB1 and IKK-α targets[146]. Long non - coding RNA FER1L4 exerts tumor suppressive

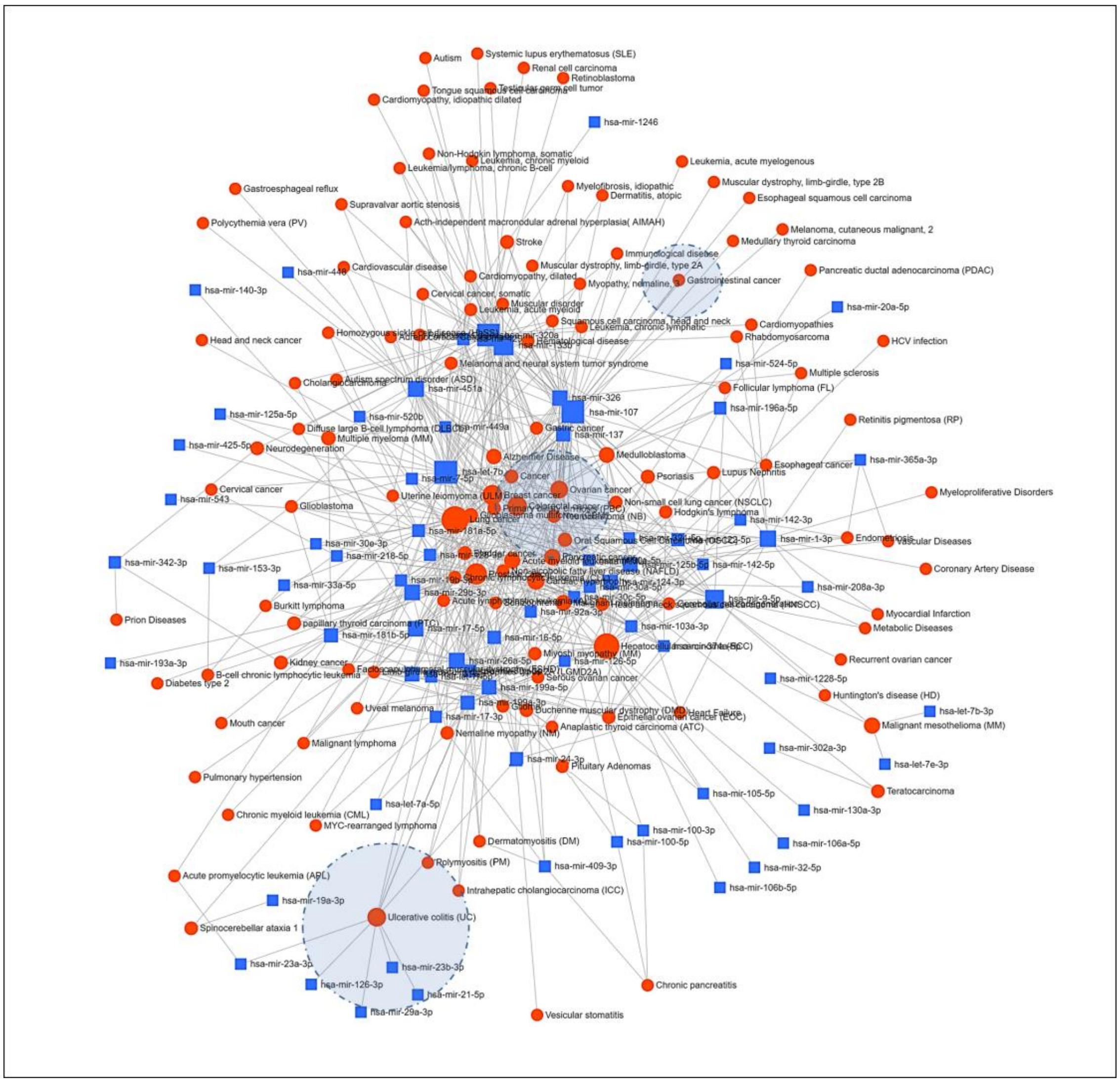


Figure 5. 12 miRNA and disease network.

effects on colon cancer by mediating miR - 106a - 5p repression[147]. The expression level of miR-106a is elevated in Intestinal biopsy, peripheral blood/serum cell of UC and CD patient [148]. mir-124 is downregulated by regulating STAT3 expression in colon tissues of pediatric patients with UC[149]. The expression level of miR-124-3p is increased in the advanced stage of CRC patients. miR - 124 - 3p work as a tumor suppressor gene in astrocytomas by targeting the repression of protein PIM1[150]. miR-1224-5p has the colitogenic ability in the gut epithelium and is directly associated with IBD disease[151]. miR-1228 is downregulated in gastric cancer tissues also overexpression of mir-1228 significantly inhibited the proliferation and colony formation of gastric cancer cells[152].

5.5.3 miRNA disease network

We used 256 our identified miRNAs to mirnetca and find out the associated miRNA-disease network[153]. 74 miRNAs are recognized with a minimum degree cutoff = 1.0 in this network out of 256 miRNAs. The layout of Fig 5.12 express the centralization of higher node degree hence the association of a disease with a significant number of miRNAs are plotted in the center. The dotted circles are the highlighted region of three IBD associated diseases i.e. Ulcerative colitis, CRC and Gastrointestinal Cancer. From the network, most of the associations are with different type of cancer e.g Lung cancer, hepatocellular carcinoma (HCC), prostate cancer, breast cancer, colorectal cancer, ovarian cancer, and pancreatic cancer. We also identified ulcerative colitis with 10 different miRNAs and gastrointestinal cancer with two different miRNAs. We found 126 different types of diseases on this network where 20% of them are associated with inflammation. Some of these diseases are Acute lymphoblastic leukemia (ALL), Chronic pancreatitis, Dermatitis, atopic, Dermatomyositis (DM), Multiple sclerosis, Polymyositis (PM), Psoriasis, etc. This also implies that inflammatory pathway analysis related to those diseases can be applied for IBD where inflammation is a common symptom

5.5.4 Disease similarity

We also evaluated the similarity of identified disease set from the miRNA-disease network to Inflammatory bowel disease. We used DisGeNET to find those disease and their corresponding gene set. The database contains gene-disease

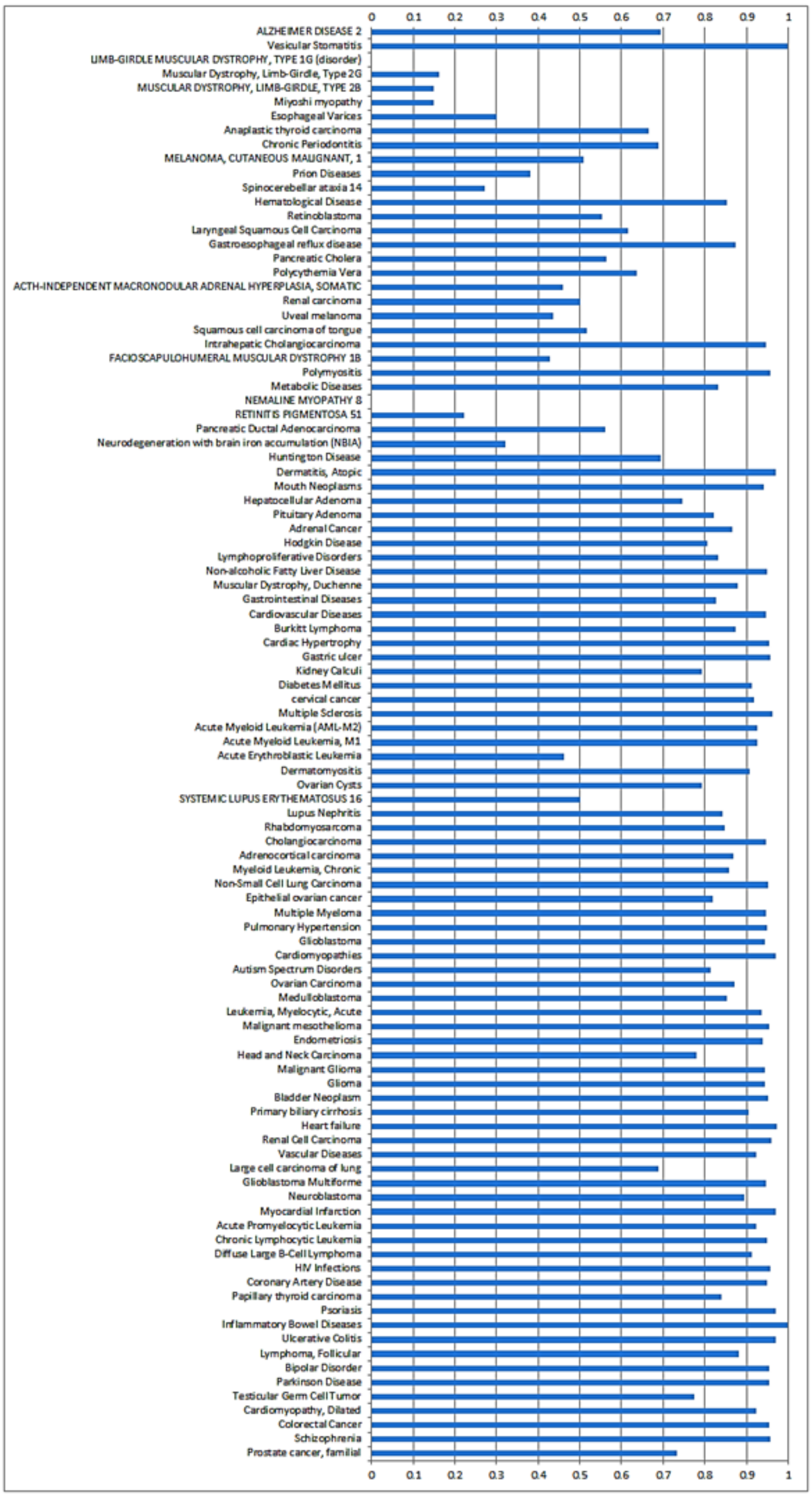


Figure 5. 13 Disease similarity between IBD and different diseases.

associations from UNIPROT, CGI, ClinGen, Genomics England, CTD (human subset), PsyGeNET, and Orphanet. We used 'DOSE' package of R and evaluated the disease similarity by using clustersim function. Figure 5.13 shows the similarity of identified disease set with the Inflammatory bowel disease. Matching score 1 indicates maximum similarity. The results show that 71% diseases are similar to IBD with a score more than 0.75. Therefore, it can be concluded that our approach is a promising method for prioritizing IBD related miRNAs and this method can be applied to other diseases.

5.6 Conclusion

Dysregulation of single or multiple miRNAs can affect normal cellular function i.e proliferation, metabolism, apoptosis, cell cycle, stem cell division, neuronal gene expression which are the major cause of different disease in human. Recent years scientists have proved the regulation of miRNAs to turn the cancer cell as malignant. Rapid discovery and progress of different clinical experiment accumulate the MRM data that can aid to discover the molecular mechanism of disease development. Our present goal was to narrow down the large domain of the multidimensional database and discover effective information. In this work, we successfully identified some important IBD related miRNAs. We also searched the literature for the association of these miRNAs to UC, CD and similar types of disease like CRC and Gastrointestinal cancer.

Chapter 6

An integrated system of clustering of simple and bipartite graph

In this work, we present new integrated software implementing the DPClusO and BiClusO algorithms to be utilized for simple and bipartite graph clustering. Our aim is to provide an open source tool with lots of user friendly options to delve into network data. This tool will provide the user with GUI based facilities for simple and bipartite graph clustering along with filtering and amalgamation of clusters, hierarchical node analysis, node distribution among cluster set and visualization of all or partial portion of a big cluster set. We name this tool DPClusSBO because this can be used to perform Density Periphery based Clustering of Simple and Bipartite graphs with Overlapping property.

6.1 Motivation and Significance

Nowadays different field of studies like system biology, economics, sociology, geography, social psychology, and business study etc. use the clustering of simple and bipartite graphs to find out the kernel information from a multivariate relational data. In system biology, analysis of PPI networks, gene expression data, gene and disease relations etc. requires different clustering methods. Cluster analysis can find different economic indicators which can reflect the development of an enterprise. In sociology, analyzing human behaviors, social impact, psychology, the ethnic interest of a demographic group is done by clustering the social network upon different metrics. In most of the above cases with or without preprocessing, the data can be represented as simple or bipartite graphs. A graph can be modelled as an adjacency matrix or an adjacency list. In most cases, practical graph data are sparse matrices. Clustering of sparse matrices is challenging and there are a few effective biclustering techniques for sparse matrices[42].

In real dataset, inherently there exist overlapping clusters which is not handled by most of the implemented algorithms found in different literatures [14][18][21][25][26]. Our algorithms provide the overlapping controlling parameter which aid to control the overlapping between generated clusters to a certain extent. We also verified the performance and effectiveness of our biclustering algorithm in terms of real and artificial dataset over some selected algorithms. Our algorithm shows the best performance in terms of three metric i.e average cluster relevancy ,average recovery rate and GO enrichment analysis[126]. Average module recovery reflects the information about how well the algorithm retrieves the actual biclusters. Average cluster relevance measures the similarity of each cluster generated by the algorithm to the actual biclusters. Rendering a large graph is a big challenge for most of the ongoing clustering software tools. Most of the clustering packages in R are with limited visualization options[33]. Besides not all software tool provides complete options like partial visualization, dynamic increasing of rendering windows, user interaction options to rearrange a drawn graph.

Our motivation is to provide a GUI based clustering tool for clustering simple and bipartite graphs where steps of data changes can be observed in textual and graphical presentation using an easy operating procedure.

6.2 Software Description

Our software consists of three modules: 1) Simple graph clustering 2) Biclustering one way 3) Biclustering two way. Simple graph clustering implements the DPClusO[11][13] algorithm. Biclustering one way and Biclustering two way implement the BiClusO[24] algorithm which was developed based on DPClusO using data folding mechanism.

Software architecture

DPClusO algorithm: DPClusO algorithm can generate overlapping clusters from a simple graph based on density and periphery tracking. If we consider a cluster, say cluster k consisting of N_k nodes then the density d_k of the cluster is expressed by the following equation:

$$d_k = \frac{E_k}{E_{k_{max}}} = \frac{2E_kN_k}{N_k(N_k - 1)} \qquad 6.1$$

Here E_{kmax} is maximum possible edges involving N_k nodes.

The cluster property is defined by

$$cp_{nk} = \frac{|E_{nk}|}{d_k \times |N_k|} \qquad 6.2$$

Here $|Enk|$ indicate the number of edges between the cluster k and a neighboring node n. The higher value of cp_{nk} implies that the node n has higher priority to be the part of cluster k.

The overlapping coefficient is defined by

$$OV = \frac{|i|^2}{|a| * |b|} \qquad (6.3)$$

The algorithm selects a node as a cluster seed from a simple graph using the highest node weight and gradually expands the cluster maintaining a user-defined density and cluster property. The algorithm proceeds by continuing node addition to the cluster and corresponding edge deletion from the original graph. Nodes which are selected upon affinity to cluster are called priority nodes.

During the creation of a cluster at each step a priority node is added and corresponding edges are removed and the residue graph is used for the subsequent step. A copy of the original graph is retained in memory. At first, the priority nodes are searched in the residue graph. If no nodes fulfill the criteria to be the part of the cluster then priority nodes are searched in the original graph. Thus the overlapping clusters are generated. Previously published papers on DPClus[11][13] and DPClusO[12][75] algorithms can be reviewed for detail explanation.

BiClusO algorithm: The details of the biclustering algorithm has been described in chapter 2.

6.3 Software architecture

Fig 6.1 shows the main architecture of DPClusSBO software using a simplified version of the class diagram. The class file *graph*, *mycluster*, *clsmatrixdata* and *clsBiCluster* are the main building block of our algorithms. DPClusO is implemented by the classes *graph* and *mycluster*. BiClusO is implemented by using the classes *graph*, *mycluster*, *clsmatrixdata* and *clsBiCluster*. User interaction like importing input file, setting different parameter, rendering cluster set and hierarchical graph is done by using the classes *frmDataLoad*, *frmDataLoadBi* and *frmDataLoadBidual*. Here *frmDataLoad* is used for clustering of simple graphs, *frmDataLoadBi* is used for biclustering(row side) and *frmDataLoadBidual* is used for 2-way biclustering (from both row side and column side).

A simple graph is represented by the class *graph* with nodes, adjacent lists of nodes and node attributes. Node attributes are the coordinate, size and color of each node which are used to render a node in the GUI frame. *Makecluster*() function is mainly used to find a cluster from a simple graph and remove corresponding edges from the graph. This function uses other different functions of the class to calculate the node weight, node degree, neighbor nodes, link and edge weight.

myCluster class uses the object of class *graph* and *region*. *Createcluster0* function in this class uses the *weightclusters()*, *degreeclusters()* functions which eventually call the *makecluster()* function to generate cluster set and save them in a vector datatype *allcluster*. According to the size of the cluster set the GUI rendering region is divided into an equal number of rectangular region and the corresponding coordinates are saved in *frmregion*. The *filtercluster()* is used to filter cluster set according to the overlapping property using equation 1 whereas *clusterjoin()* function is used to join cluster set.

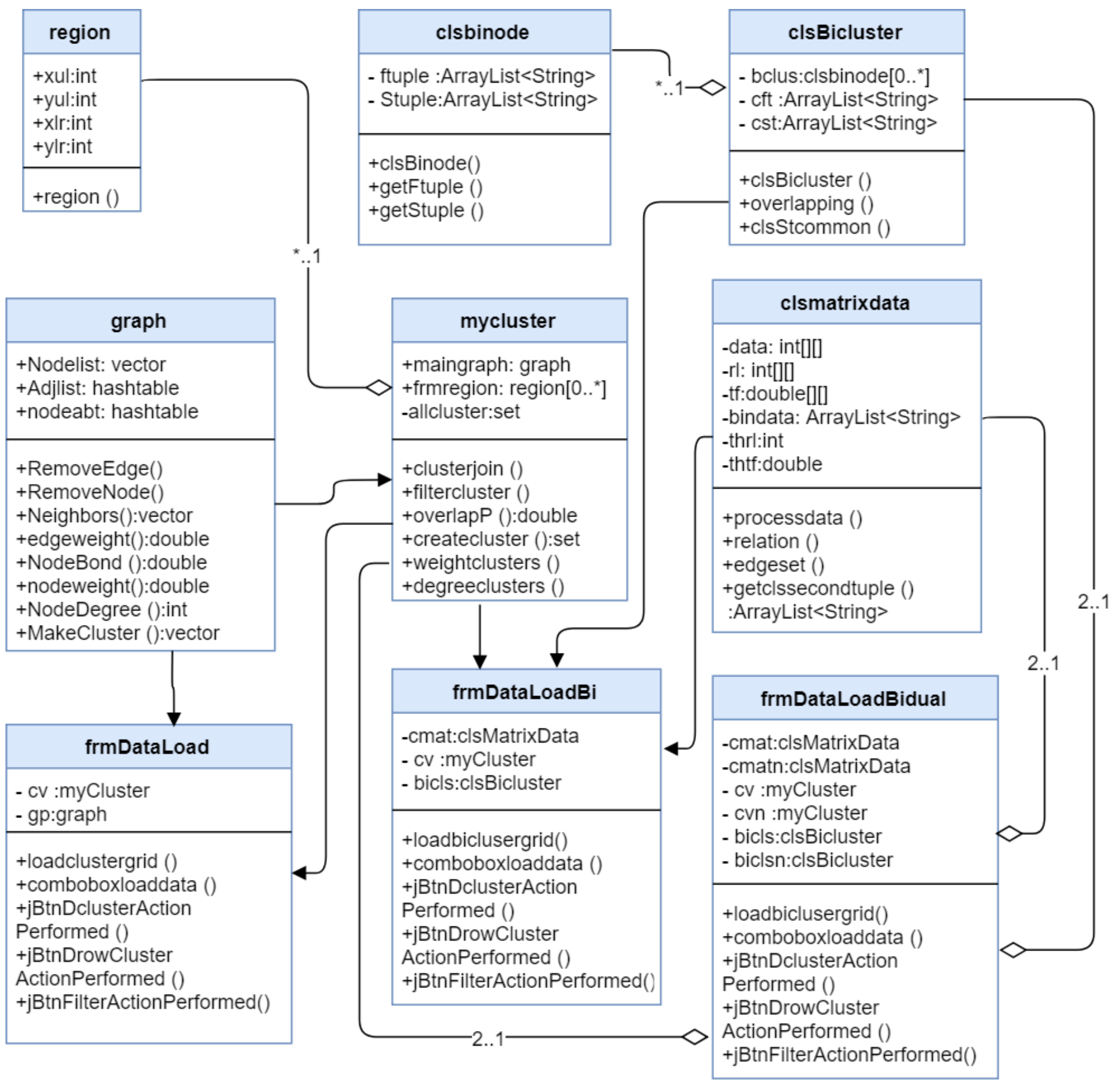


Figure 6. 1 Class diagram of DPClusSBO.

clsmatrixdata class takes a binary input matrix and generates the Tanimoto coefficient matrix (tf) and relation number matrix (rl). The *processdata()* function uses the threshold points of Tanimoto coefficient (thtf) and relation number (thrl) as filtering parameters to filter the matrix tf and rf. After filtering, a tab separated string of arraylist i.e *bindata* representing the binary relation between the row names of the input matrix is saved in *bindata*. This arraylist is provided as input to *myCluster* class to create cluster set.

The function *getclssecondtuple()* in *clsmatrixdata* class generates the second node set of any cluster using their attachment probability. The class *clsBiCluster* has constructors which take those clusters, attachment probability and object of *clsmatrixdata* as input and assign individual bicluster to a hash map of the object of the class *clsbinode*.

In the case of two way biclustering, two sets of binary relational data, one for row names and another for columns name are generated. *cv* and *cvn* is the generated cluster sets for two types of data.

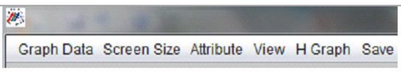
User interaction for clustering of simple graphs, biclustering(row side) and biclustering (both row side and column side) is done through *frmDataLoad , frmDataLoadBi and frmDataLoadBidual* class.

6.4 Software functionality

Two different types of functionality, graphics and text-based have been implemented in analyzing the three types of clustering. Filtering cluster number according to different size and serial number allows the user to fit cluster set in screen size and maximize the visualization. Following a series of consecutive function execution generates the text-based data changes and finally the GUI based rendering of the cluster set.

6.4.1 Simple clustering

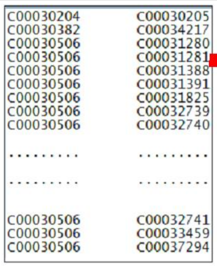
Loading Input Network: The simplest way to represent a network is the edge list format and this tool takes the input network represented by this format. Data should be provided by tab separated text file without header information as shown in



a) Main menu

Graph Data -> File Load
Screen Size -> Default, Moderate, Large
Attribute -> Edge Property, Font Property, Back Color , Unique Color Node
View -> Circular, Random, Circular Random
H Graph -> All Overlapping, Neighbor Overlapping
Save -> Multiple Cluster, Single Cluster, Hierarchical Graph, Hierarchical Node, Adjacent List, Cluster Info, Hierarchical Graph info , Node Distribution

b) Sub menus



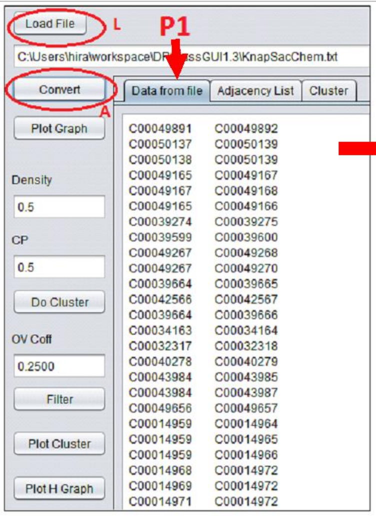
```
C00030204    C00030205
C00030382    C00034217
C00030506    C00031280
C00030506    C00031281
C00030506    C00031388
C00030506    C00031391
C00030506    C00031825
C00030506    C00032739
C00030506    C00032740

.........    .........

.........    .........

C00030506    C00032741
C00030506    C00033459
C00030506    C00037294
```

c) Input text file

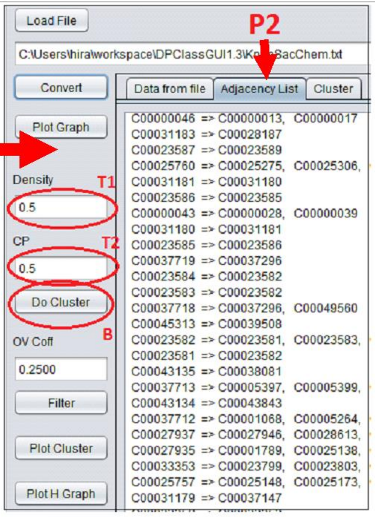


C:\Users\hira\workspace\DP[illegible]ssGUI1.3\KnapSacChem.txt

Density: 0.5
CP: 0.5
OV Coff: 0.2500

```
C00049891   C00049892
C00050137   C00050139
C00050138   C00050139
C00049165   C00049167
C00049167   C00049168
C00049165   C00049166
C00039274   C00039275
C00039599   C00039600
C00049267   C00049268
C00049267   C00049270
C00039664   C00039665
C00042566   C00042567
C00039664   C00039666
C00034163   C00034164
C00032317   C00032318
C00040278   C00040279
C00043984   C00043985
C00043984   C00043987
C00049656   C00049657
C00014959   C00014964
C00014959   C00014965
C00014959   C00014966
C00014968   C00014972
C00014969   C00014972
C00014971   C00014972
```

d) Data load

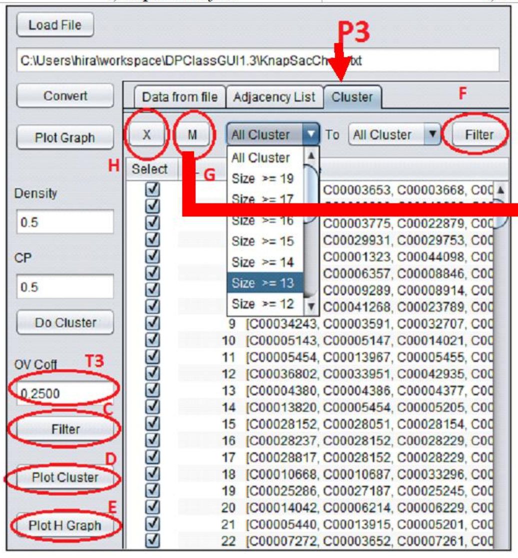


C:\Users\hira\workspace\DPClassGUI1.3\K[illegible]SacChem.txt

Density: 0.5
CP: 0.5
OV Coff: 0.2500

```
C00000046 => C00000013,  C00000017
C00031183 => C00028187
C00023587 => C00023589
C00025760 => C00025275,  C00025306,
C00031181 => C00031180
C00023586 => C00023585
C00000043 => C00000028,  C00000039
C00031180 => C00031181
C00023585 => C00023586
C00037719 => C00037296
C00023584 => C00023582
C00023583 => C00023582
C00037718 => C00037296,  C00049560
C00045313 => C00039508
C00023582 => C00023581,  C00023583,
C00023581 => C00023582
C00043135 => C00038081
C00037713 => C00005397,  C00005399,
C00043134 => C00043843
C00037712 => C00001068,  C00005264,
C00027937 => C00027946,  C00028613,
C00027935 => C00001789,  C00025138,
C00033353 => C00023799,  C00023803,
C00025757 => C00025148,  C00025173,
C00031179 => C00037147
```

e) Adjacency list conversion

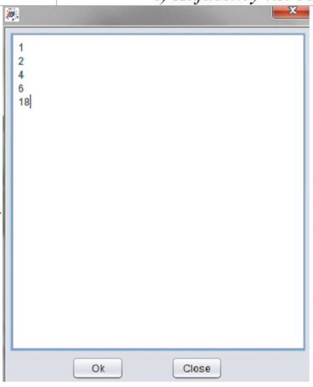


C:\Users\hira\workspace\DPClassGUI1.3\KnapSacCh[illegible]txt

Density: 0.5
CP: 0.5
OV Coff: 0.2500

```
     C00003653, C00003668, C0(
     C00003775, C00022879, C0(
     C00029931, C00029753, C0(
     C00001323, C00044098, C0(
     C00006357, C00008846, C0(
     C00009289, C00008914, C0(
     C00041268, C00023789, C0(
 9  [C00034243, C00003591, C00032707, C0(
10  [C00005143, C00005147, C00014021, C0(
11  [C00005454, C00013967, C00005455, C0(
12  [C00036802, C00033951, C00042935, C0(
13  [C00004380, C00004386, C00004377, C0(
14  [C00013820, C00005454, C00005205, C0(
15  [C00028152, C00028051, C00028154, C0(
16  [C00028237, C00028152, C00028229, C0(
17  [C00028817, C00028152, C00028229, C0(
18  [C00010668, C00010687, C00033296, C0(
19  [C00025286, C00027187, C00025245, C0(
20  [C00014042, C00006214, C00006229, C0(
21  [C00005440, C00013915, C00005201, C0(
22  [C00007272, C00003652, C00007261, C0(
```

f) Cluster conversion



```
1
2
4
6
18
```

g) Multiple filtering

Figure 6. 2 Main menu, sub menus and different operational buttons of Simple clustering.

Fig. 6.2c. Each line in the text file represents an edge in term of the pair of nodes it connects. By clicking sub menu item "File Load" from menu Item "Graph Data" (Fig 6.2b), data load form will appear. User has to click "Load File" button (Fig 6.2d, "L") to open file browser to select the desired data file. After loading from text file, data will be shown in "Data from file" tabbed pane (Fig 6.2d "P1"). For illustrating the Simple clustering, we are using a structural similarity network of metabolites collected from KNApSAcK database consisting of 9129 nodes and 10,000 edges

Conversion to Adjacency List: Clicking "Convert" button (Fig 6.2d, "A") will convert the network from edge list format to adjacency list format. Adjacency list data will be shown in "Adjacency List" tabbed pane (Fig 6.2e "P2").

Clustering: User has to click "Do Cluster" button (Fig 6.2e,"B") to create the clusters. Clusters will be created and populated in the "Cluster" tabbed pane (Fig 6.2f,"P3") on descending order of their size. By default a cluster number is given with integer value beginning from one to total cluster number. These numbers are essential for custom filtering of the clusters.

6.4.2 Simple Cluster Filtering

The numbers of sub graphs of a given density are too many in networks of reasonable size [12]. The algorithm of Georgii et al. [154] attempted to find all possible sub graphs of a given density, but the results indicate that the number of clusters as well as computational time would be very high and would not be feasible for big and dense networks. Therefore DPClusO heuristically try to assign each node to clusters as large as possible and also avoid generating too many nearly identical overlapping clusters to narrow down the candidate clusters. But still the number of clusters generated by DPClusO is relatively large though each of these high-density clusters can be considered as cohesive group of entities. However users may be interested in small number of clusters with limited overlapping and of desired sizes. Therefore, we provide some filtering options with this tool. Filtering can extract a subset of clusters from created cluster set. Big cluster set hinder the visualization of clusters.

Filtering options allow user to visualize any portion of created clusters according to following settings.

Overlapping Coefficient: User can enter decimal value within the following range: *0 <=ov<= 1* in "OV Coff" text field (Fig 6.2f, "T3") and click "Filter" button (Fig 6.2f, "C") to reduce the cluster set according to overlapping coefficient. After filtering, no two clusters are more overlapped than the entered OV coefficient. Entering '0' in the "OV Coff" text field will allow no overlapping between any two clusters.

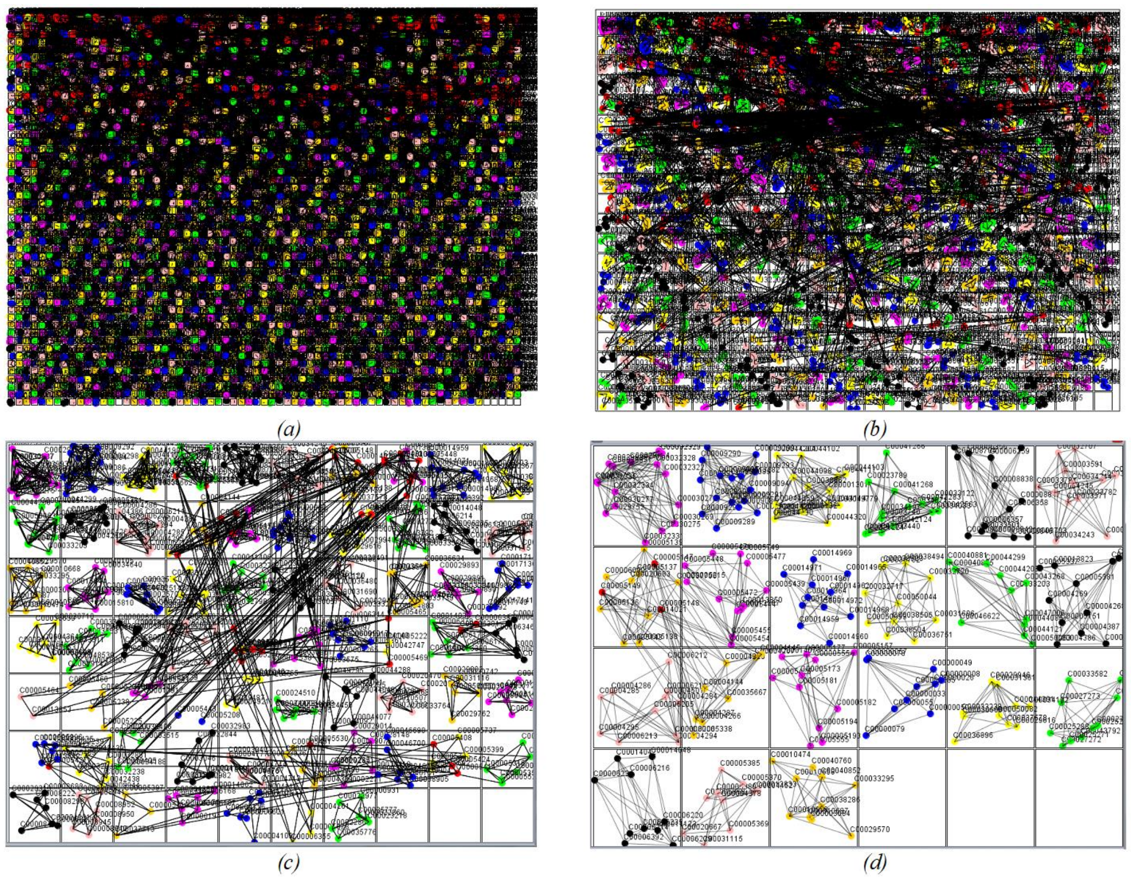


Figure 6. 3 Filtering and visualizing. (a) Without any filter, total 3915 clusters. (b) Filtering with OV coff = 0.25 and cluster size >=4, total 537 clusters. (c) Filtering with OV coff = 0.25 and cluster size >=7, total 67 clusters. (d) Filtering with OV coff = 0.25 and cluster size >=10, total 21 clusters.

Cluster Size: In the tabbed pane "Cluster", two combo boxes always hold distinct cluster size by descending order. User can enter size range from these two combo boxes to filter clusters of specific sizes.

Cluster Number: User can select any cluster by unchecking the left column of data table and click the cross signed button (Fig 6.2f,"H") to remove the cluster.

Multiple Cluster Number: Right beside the cross sign button the "M" button (Fig 6.2f,"G") is used to select multiple clusters. A text box window will appear by clicking the button "M". User can enter desired cluster IDs one by one in a line by ascending order.

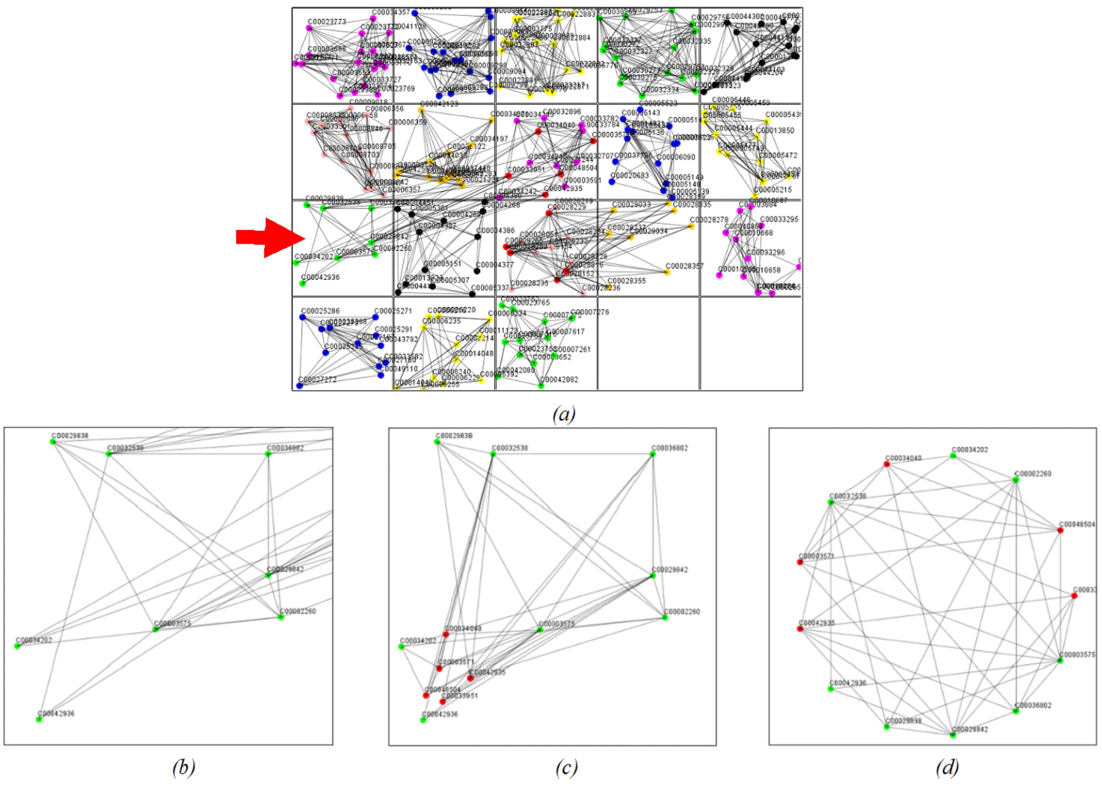


Figure 6. 4 (a) Cluster generated with density=0.5, cluster property =0.5 and filtered using OV coff = 0.35, size>=12. Total 18 clusters. (b) Magnified view of cluster 11. (c) Extracted overlapping nodes. (d) Circular layout of cluster 11.

Once clicking the "Filter" button (Fig 6.2f, "F") all clusters except the desired clusters (Fig 6.2g) will be unchecked. User can then click the cross button to remove all undesired clusters.

6.4.3 Simple Cluster Visualization

Cluster visualization is a technique to represent sub graphs from a network data by using graphical tools. In order to perceive cluster and their interconnectivity visually from big network data this representation is very useful.

Plot Cluster: After the creation of clusters, visualization can be done by clicking "Plot Cluster" button (Fig 6.2f, "D") (Fig 6.3). A new GUI window will open on right side with grid layout indicating the cluster boundary. Basically view window is divided by grid layout. For visualization of N clusters, the viewing area is divided into x columns and y rows by solving $xy=N$ and $x:y=4:3$ equations and by rounding or truncating x and y to nearest integers. Thus if total cluster is 48 then grid layout will have 8 columns and 6 rows. If it is 20 then grid layout will have 5 columns and 4 rows. The same ratio is maintained even if the screen size is changed by user. We draw nodes of each cluster with distinct color however all overlapping nodes are represented with red color and each overlapping node is drawn only once (Fig 6.3). The clusters are arranged row wise according to descending order of size. To aid to vary visualization properties several different options are added via menu items "Screen Size", "Attribute" and "View" (Fig 6.2b). Below we briefly discuss these options.

(i) *Screen Size:* Three different types of screen size are included here i.e. default, moderate and large. User can select the screen size by clicking the appropriate size.

(ii) *Attribute:* This menu item has four different options to make the visualization user friendly so that user can easily interpret each node and edge in a cluster as follows.

a) "Edge" property option set the different color level of edges. With large number of

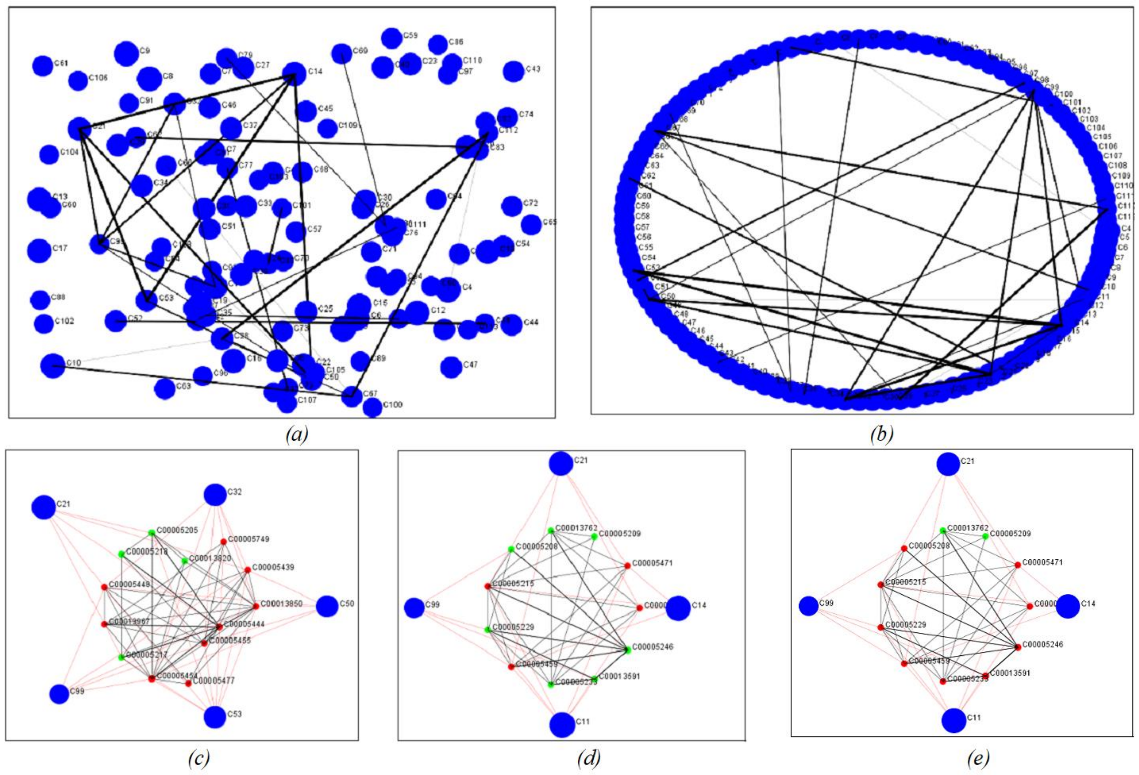


Figure 6. 5 Hierarchical graph of clusters using cluster no 4 to 113 by filtering with size >=8 and <=16. Clusters have been generated with density=0.5, cluster property =0.5 and without any filtering by OV coff. (a) Random drawing. (b) Circular clockwise drawing with descending order of cluster size. (c) Magnified view of cluster no 14. (d) Cluster no 32 with only neighbor overlapping nodes indicated by red. (e) Cluster no 32 with all overlapping nodes.

edges between a cluster and its neighbors make the graph a little bit crowded. Making different choice from "deep" to "zero" will give better viewing of the graph.

b) "Font" property allow user to set the font of the node label. By selecting the font name, font size and font type user can set the font property of a node label.

c) "Back Color" property allows user to set the background color of GUI window with two different color types (i.e. white, button face).

d) "Unique Node color" property allows user to set same color for the nodes of a cluster in magnifying window.

(iii) *View:* This menu item has three different options to place the node position in a cluster region with different coordinate patterns as follows.
a) "Random" pattern set the node coordinates of a cluster in random way using the rectangular height, width of GUI grid window.
b) "Circular" pattern set the node coordinates of a cluster using an elliptic equation whose axes fit rectangular GUI window of the cluster with maximum possible size. All node coordinates are fitted to the periphery of the ellipse.
c) "Circular Random" pattern combines the effect of random axis length and an elliptic equation to set coordinates back and forth on the periphery of the ellipse.

Single Cluster Visualization: User can magnify and make better view of a single cluster by double clicking the grid region of the cluster. A new window will open with magnified view of the grid (Fig 6.4b*)*. It is very important to open single cluster window to view all nodes of a cluster. First double click on the magnified grid will extract all remaining overlapping nodes of the selected cluster from its overlapping neighbors (Fig 6.4c) and second double click will rearrange the nodes according to different view attributes (Fig 6.4d).
Simple cluster Hierarchical Graph: User can visualize the hierarchical graph by clicking the "Plot H Graph" button (Fig. 2f, "E"). In a hierarchical Graph each cluster is considered as a node. Node radius is calculated by formula $r = Log(N)*10$ where N is total number of nodes in a cluster. The edges between non overlapping nodes of two clusters are considered to determine edge thickness between hierarchical graph nodes. In GUI window, edge line thickness is calculated by formula $d = Log(E)*10$ where E is total number of edges between non overlapping nodes of two clusters. Also by selecting different visualization options from menu user can make the GUI

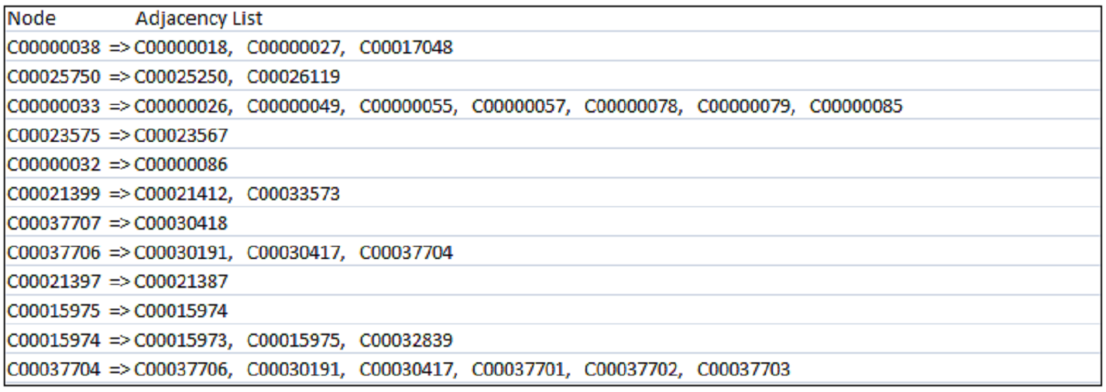
window more suitable. In order to arrange the nodes of the hierarchical graph according their size user has to select the "Circular" option from view menu and click the "Plot H Graph" button (Fig 6.2f, "E"). All nodes of hierarchical Graph will be arranged according to their size in circular fashion (Fig 6.5b). User can drag and arrange the nodes in view window. An individual node of hierarchical graph can be visualized by double clicking on the node. A new GUI window will appear with actual view of selected node and neighbor cluster nodes (Fig 6.5c). Red color edges indicate the relation between adjacent clusters and nodes of selected cluster. These relations are edges between all non overlapping nodes. Relation line thickness is kept constant between big blue node and small actual nodes even if the possibility of multiple edges from a small actual node to big blue node. All green nodes indicate non overlapping nodes whereas red nodes are overlapping nodes. Red nodes can be further divided into two groups by selecting two different options from hierarchical graph menu. Selecting "Neighbor Overlapping", the nodes which are overlapped with respect to adjacent neighbor in hierarchical graph will be colored by red (Fig 6.5d). As hierarchical graph are drawn considering the edges between non overlapping nodes of the clusters of original graph there may be a possibility that two cluster have some overlapping node but no edges between non overleaping nodes. Taken consideration of the fact, "All Overlapping" sub menu item is added to the "H graph" menu (Fig 6.2b). Selection of this sub menu item will make node color red which are overlapped to any cluster in the graph along with neighbor clusters (Fig 6.5e).

6.4.4 Brief Discussion of Metabolite Clusters

We successfully generated clusters using a structural similarity network of metabolites collected from KNApSAcK database using density 0.6, cluster property 0.5 and filtered using overlapping coefficient 0.05. Total 2714 clusters were found with minimum size 2 to maximum size 16. After filtering with minimum size 12 we found biggest 6 clusters. Supplementary figure 2 shows metabolites included in top six cluster groups. Cluster 1 contains 16 elements which are all of chemical class

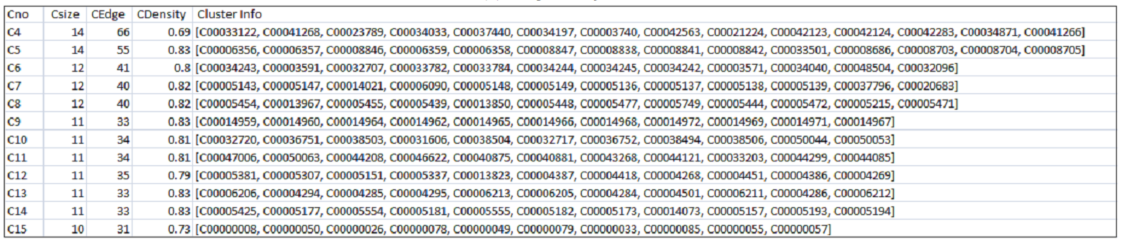
"Triterpenoid saponins". Cluster 2 contains 16 elements which are all of chemical class "Proanthocyanidins". Cluster 3 contains 15 elements which are all of chemical class "Acetogenin". Cluster 4 contains 14 elements which are all of chemical class

6.4.5 Simple cluster output Files



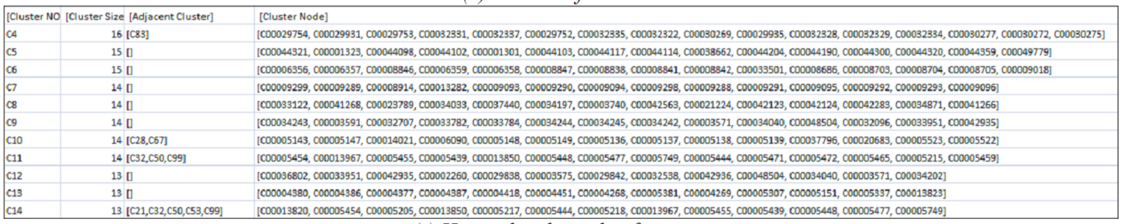
Node	Adjacency List
C00000038 =>	C00000018, C00000027, C00017048
C00025750 =>	C00025250, C00026119
C00000033 =>	C00000026, C00000049, C00000055, C00000057, C00000078, C00000079, C00000085
C00023575 =>	C00023567
C00000032 =>	C00000086
C00021399 =>	C00021412, C00033573
C00037707 =>	C00030418
C00037706 =>	C00030191, C00030417, C00037704
C00021397 =>	C00021387
C00015975 =>	C00015974
C00015974 =>	C00015973, C00015975, C00032839
C00037704 =>	C00037706, C00030191, C00030417, C00037701, C00037702, C00037703

(a) Adjacency list



Cno	Csize	CEdge	CDensity	Cluster Info
C4	14	66	0.69	[C00033122, C00041268, C00023789, C00034033, C00037440, C00034197, C00003740, C00042563, C00021224, C00042123, C00042124, C00042283, C00034871, C00041266]
C5	14	55	0.83	[C00006356, C00006357, C00008846, C00006359, C00006358, C00008847, C00008838, C00008841, C00008842, C00033501, C00008686, C00008703, C00008704, C00008705]
C6	12	41	0.8	[C00034243, C00003591, C00032707, C00033782, C00033784, C00034244, C00034245, C00034242, C00003571, C00034040, C00048504, C00032096]
C7	12	40	0.82	[C00005143, C00005147, C00014021, C00006090, C00005148, C00005149, C00005136, C00005137, C00005138, C00005139, C00037796, C00020683]
C8	12	40	0.82	[C00005454, C00013967, C00005455, C00005439, C00013850, C00005448, C00005477, C00005749, C00005444, C00005472, C00005215, C00005471]
C9	11	33	0.83	[C00014959, C00014960, C00014964, C00014962, C00014965, C00014966, C00014968, C00014972, C00014969, C00014971, C00014967]
C10	11	34	0.81	[C00032720, C00036751, C00038503, C00031606, C00038504, C00032717, C00036752, C00038494, C00038506, C00050044, C00050053]
C11	11	34	0.81	[C00047006, C00050063, C00044208, C00046622, C00040875, C00040881, C00043268, C00044121, C00033203, C00044299, C00044085]
C12	11	35	0.79	[C00005381, C00005307, C00005151, C00005337, C00013823, C00004387, C00004418, C00004268, C00004451, C00004386, C00004269]
C13	11	33	0.83	[C00006206, C00004294, C00004285, C00004295, C00006213, C00006205, C00004284, C00004501, C00006211, C00004286, C00006212]
C14	11	33	0.83	[C00005425, C00005177, C00005554, C00005181, C00005555, C00005182, C00005173, C00014073, C00005157, C00005193, C00005194]
C15	10	31	0.73	[C00000008, C00000050, C00000026, C00000078, C00000049, C00000079, C00000033, C00000085, C00000055, C00000057]

(b) Cluster information



[Cluster NO	[Cluster Size	[Adjacent Cluster]	[Cluster Node]
C4	16	[C83]	[C00029754, C00029931, C00029753, C00032331, C00032337, C00029752, C00032335, C00032322, C00030269, C00029935, C00032328, C00032329, C00032334, C00030277, C00030272, C00030275]
C5	15	[]	[C00044321, C00001323, C00044098, C00044102, C00001301, C00044103, C00044117, C00044114, C00038662, C00044204, C00044190, C00044300, C00044320, C00044359, C00049779]
C6	15	[]	[C00006356, C00006357, C00008846, C00006359, C00006358, C00008847, C00008838, C00008841, C00008842, C00033501, C00008686, C00008703, C00008704, C00008705, C00009018]
C7	14	[]	[C00009299, C00009289, C00008914, C00013282, C00009093, C00009290, C00009094, C00009298, C00009288, C00009291, C00009095, C00009292, C00009293, C00009096]
C8	14	[]	[C00033122, C00041268, C00023789, C00034033, C00037440, C00034197, C00003740, C00042563, C00021224, C00042123, C00042124, C00042283, C00034871, C00041266]
C9	14	[]	[C00034243, C00003591, C00032707, C00033782, C00033784, C00034244, C00034245, C00034242, C00003571, C00034040, C00048504, C00032096, C00033951, C00042935]
C10	14	[C28,C67]	[C00005143, C00005147, C00014021, C00006090, C00005148, C00005149, C00005136, C00005137, C00005138, C00005139, C00037796, C00020683, C00005523, C00005522]
C11	14	[C32,C50,C99]	[C00005454, C00013967, C00005455, C00005439, C00013850, C00005448, C00005477, C00005749, C00005444, C00005471, C00005472, C00005465, C00005215, C00005459]
C12	13	[]	[C00036802, C00033951, C00042935, C00002260, C00029838, C00003575, C00029842, C00032538, C00042936, C00048504, C00034040, C00003571, C00034202]
C13	13	[]	[C00004380, C00004386, C00004377, C00004387, C00004418, C00004451, C00004268, C00005381, C00004269, C00005307, C00005151, C00005337, C00013823]
C14	13	[C21,C32,C50,C53,C99]	[C00013820, C00005454, C00005205, C00013850, C00005217, C00005444, C00005218, C00013967, C00005455, C00005439, C00005448, C00005477, C00005749]

(c) Hierarchical graph information

Figure 6. 6 Different text file from simple clustering.

"Triterpene". Cluster 5 contains 14 elements which are all of chemical class "Flavanonol". Cluster 6 contains 12 elements which are all of chemical class "Saponin". These results indicate that Simple clustering successfully clustered structurally similar metabolites, also it can be concluded that structurally similar metabolites belong to specific types and functions which fact is supported by other previous studies [17][155][156].

Menu option "Save" provides the facility of saving different information to image files or text files (Fig. 2b). Eight different types of output files with four types of image files and four types of text files can be created by user choice. Before saving an image user can rearrange the nodes of a graph by mouse dragging. Also edge color setting, node font setting from menu will give the image a clear and aesthetic view.

Multiple Cluster: "Multiple Cluster" (Fig 6.2b) sub menu item is used to create "Multiple_cluster.png" image file of the cluster generated by Simple clustering. User has to activate the GUI window of cluster by clicking over the window and then click "Multiple Cluster" sub menu item to save the image. File save option with default name "Multiple_cluster.png" will appear. User can change the file name also. Once the save button is clicked, graphics from selected window will be saved to the image file.

Single Cluster: "Single cluster" (Fig 2b) sub menu item allow user to save the image of magnified single cluster window. User can activate the desired window by clicking over it and by selecting the "Single Cluster" sub menu item to save the image. The default filename in this case is "Single_cluster.png".

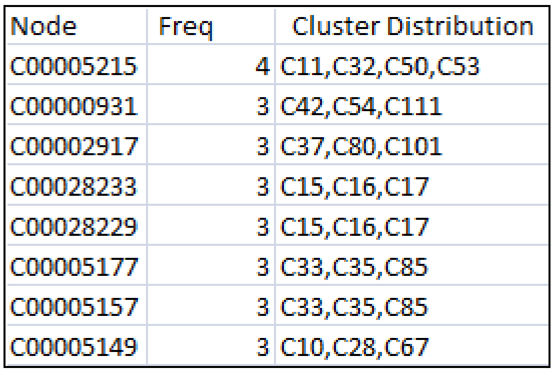
Hierarchical Graph: GUI view of hierarchical graph of a given cluster set can be saved in an image file with default name "Hierarchical_Graph.png". User should activate the window of hierarchical graph and select this sub menu item to save the image.

Hierarchical Node: Detail view of a node from hierarchical graph window can be saved by using this sub menu item. Once user creates the detail view by double clicking the node from hierarchical graph and activate the window, image can be saved by clicking this sub menu item.

Adjacency List: This sub menu item is used to save the adjacency list of the graph in a text file. After being created from binary relational input text file the adjacency list populate the tabbed pane labeled by "Adjacency List". By clicking this sub menu item user can save the data with following format: First column indicates individual nodes. A comma separated list followed by "=>" character in the second column is adjacency list of the corresponding nodes (Fig 6.6a)

Cluster Info: After creating and filtering the clusters, user can save detail cluster information by clicking this sub menu item. A text file will be generated with following format (Fig 6.6b). First column indicates the cluster ID, second column cluster size (total nodes), third column total edges, forth column cluster density and fifth column shows cluster node labels.

Hierarchical Graph Info: After creating the hierarchical graph by clicking the button "Plot H Graph" this sub menu item will allow saving hierarchical graph information. Default name for this file is "ClusterHierarchyinfo.txt", according to the following format; first column indicates the cluster no, second column the cluster size, third column adjacent cluster list and fourth column shows cluster node list (Fig 6.6c).



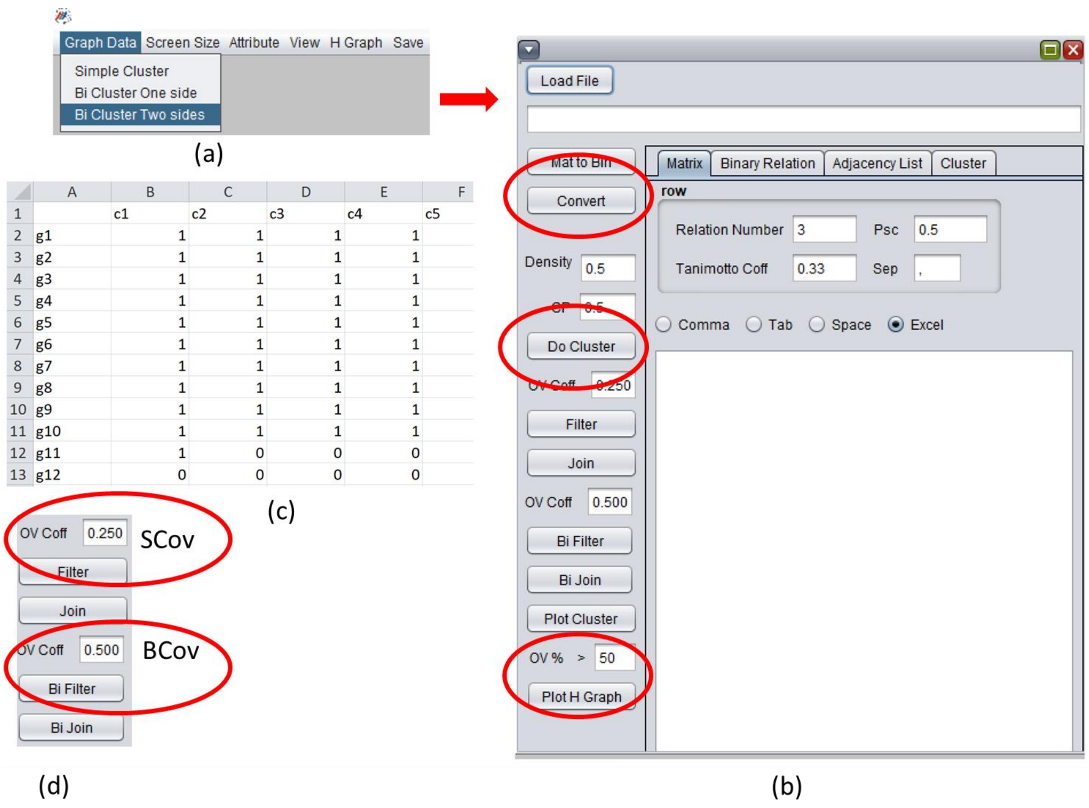
Node	Freq	Cluster Distribution
C00005215	4	C11,C32,C50,C53
C00000931	3	C42,C54,C111
C00002917	3	C37,C80,C101
C00028233	3	C15,C16,C17
C00028229	3	C15,C16,C17
C00005177	3	C33,C35,C85
C00005157	3	C33,C35,C85
C00005149	3	C10,C28,C67

Figure 6. 7 Distribution of nodes to clusters.

Node Distribution: An individual node and the clusters to which it belongs are arranged in a text file (Fig 6.7). User can save this file with default name "NodeDistribution.txt", according to the following format; first column indicates the edge label, second column the frequency of the node occurrence and third column shows the set of cluster IDs.

6.4.6 Biclustering

The procedure for generating bicluster one-way and two-way is same hence we only describe here only two-way biclustering.



Density 0.5

CP 0.5

OV Coff 0.250

OV Coff 0.500

OV % > 50

Relation Number 3

Psc 0.5

Tanimotto Coff 0.33

Sep ,

Excel [✓]

OV Coff 0.250

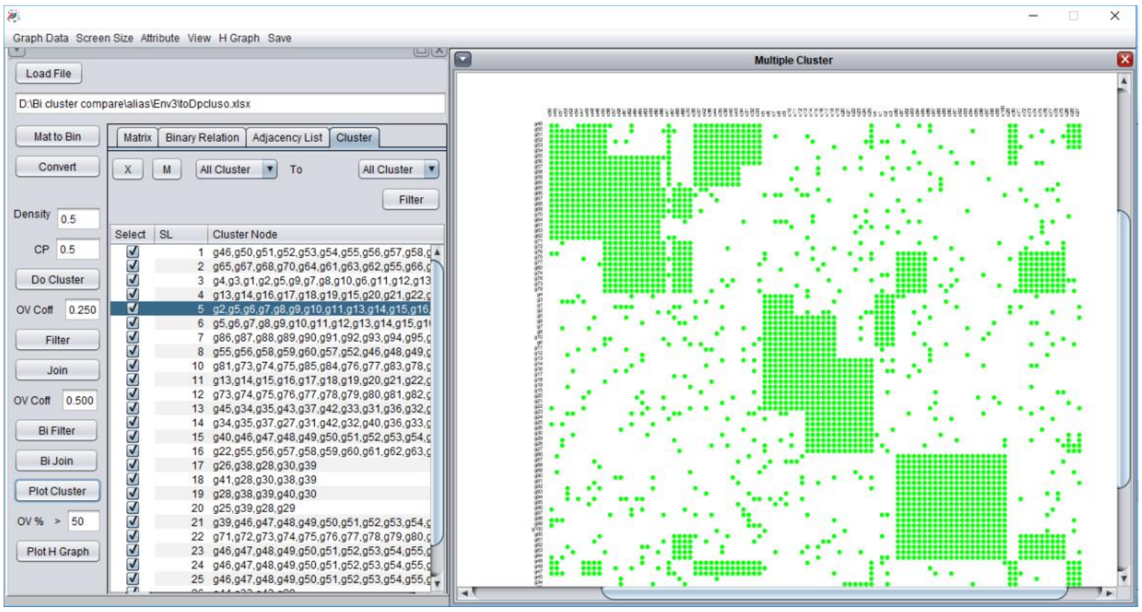
OV Coff 0.500

	A	B	C	D	E	F
1		c1	c2	c3	c4	c5
2	g1	1	1	1	1	
3	g2	1	1	1	1	
4	g3	1	1	1	1	
5	g4	1	1	1	1	
6	g5	1	1	1	1	
7	g6	1	1	1	1	
8	g7	1	1	1	1	
9	g8	1	1	1	1	
10	g9	1	1	1	1	
11	g10	1	1	1	1	
12	g11	1	0	0	0	
13	g12	0	0	0	0	

Figure 6. 8 Creating BiClustering from a matrix.

The difference between one way and two-way biclustering is that one-way bi clustering considers only row-wise comparison to generate a simple graph whereas two-way bi clustering considers both row and column. In this example we used a gene vs. condition binary (0,1) synthetic dataset by embedding specific number of biclusters where a 1 represents a differentially expressed gene under a specific condition. Furthermore, we randomly inserted 1s as noise at 9% of the non-cluster region of the dataset

Loading Input data: The input file should be an excel file with the appropriate row and column label or two column tab, comma or space separated text file representing a bipartite graph (Fig 6.8c).



D:\Bi cluster compare\alias\Env3\toDpcluso.xlsx

Density 0.5
CP 0.5
OV Coff 0.250
OV Coff 0.500
OV % > 50

All Cluster To All Cluster

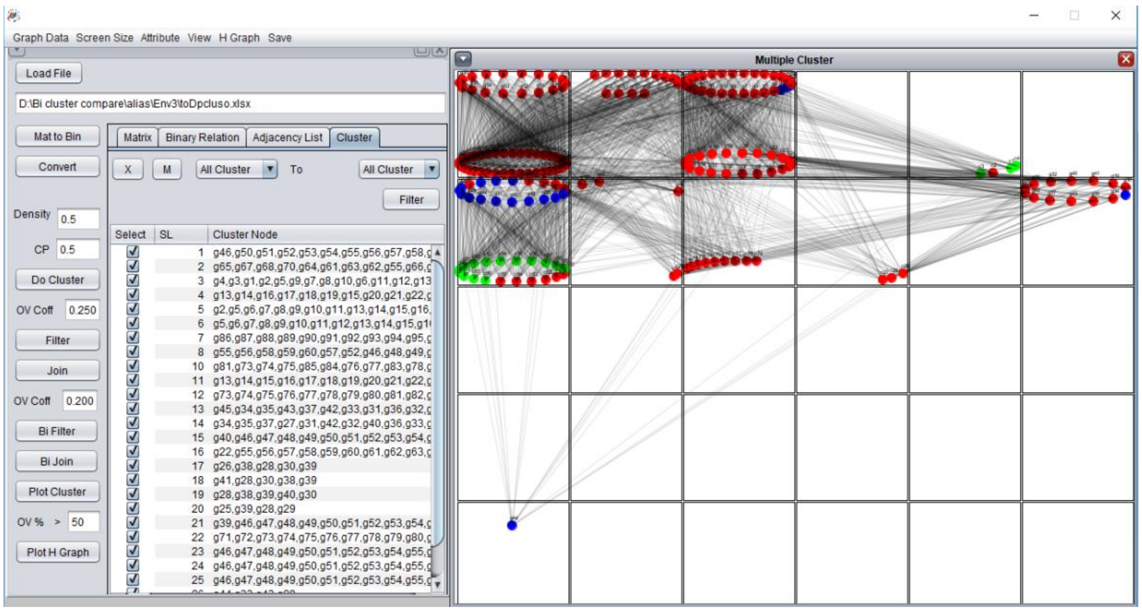
Select	SL	Cluster Node
✓	1	g46,g50,g51,g52,g53,g54,g55,g56,g57,g58,
✓	2	g65,g67,g68,g70,g64,g61,g63,g62,g55,g66,
✓	3	g4,g3,g1,g2,g5,g9,g7,g8,g10,g6,g11,g12,g13
✓	4	g13,g14,g16,g17,g18,g19,g15,g20,g21,g22,
✓	5	g2,g5,g6,g7,g8,g9,g10,g11,g13,g14,g15,g16,
✓	6	g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g1
✓	7	g86,g87,g88,g89,g90,g91,g92,g93,g94,g95,
✓	8	g55,g56,g58,g59,g60,g57,g52,g46,g48,g49,
✓	10	g81,g73,g74,g75,g85,g84,g76,g77,g83,g78,
✓	11	g13,g14,g15,g16,g17,g18,g19,g20,g21,g22,
✓	12	g73,g74,g75,g76,g77,g78,g79,g80,g81,g82,
✓	13	g45,g34,g35,g43,g37,g42,g33,g31,g36,g32,
✓	14	g34,g35,g37,g27,g31,g42,g32,g40,g36,g33,
✓	15	g40,g46,g47,g48,g49,g50,g51,g52,g53,g54,
✓	16	g22,g55,g56,g57,g58,g59,g60,g61,g62,g63,
✓	17	g26,g38,g28,g30,g39
✓	18	g41,g28,g30,g38,g39
✓	19	g28,g38,g39,g40,g30
✓	20	g25,g39,g28,g29
✓	21	g39,g46,g47,g48,g49,g50,g51,g52,g53,g54,
✓	22	g71,g72,g73,g74,g75,g76,g77,g78,g79,g80,
✓	23	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,
✓	24	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,
✓	25	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,

Figure 6. 9 Matrix view of a bicluster set.

We show the example of using an excel file. Row and column label in the excel file represent the two node sets of a bipartite graph. 1 represents an edge between the corresponding row label and column label. A "0" must be present if there is no edge between row label and column label. Click "load file" and select the excel file. Click "Mat to bin" to convert the data as a simple graph. Click "Convert" to convert the simple graph as an adjacency matrix. Click "do cluster" to generate biclusters. Density =0.5, *Cp*(Cluster property)=0.5 and *OV* (Overlapping) = 0.5 are default parameter for optimal cluster generation from simple graphs[]. Two additional default parameter Tanimoto coefficient =0.33 and relation number = 3 are needed to generate biclusters.

Filter/ Join: There are two coefficients to filter or join biclusters 1) DpClusO overlapping coefficient 2) Bi Cluster overlapping coefficient. User can use any one or both of them to filter and join biclusters.

View bicluster: There are two types of view to render the bicluster set which are Bicluster Graph view and Bicluster Matrix view. User can select any types of view from the view menu and click the plot cluster to render the bicluster set.



D:\Bi cluster compare\alias\Env3\toDpcluso.xlsx

Density 0.5

CP 0.5

OV Coff 0.250

OV Coff 0.200

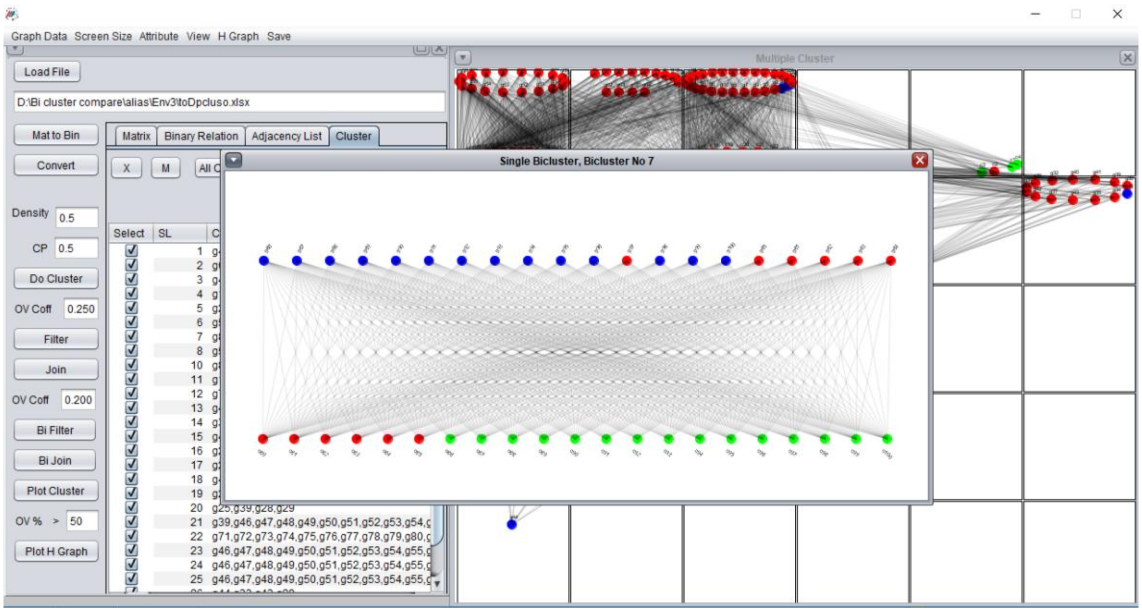
OV % > 50

Select	SL	Cluster Node
✓	1	g46,g50,g51,g52,g53,g54,g55,g56,g57,g58,
✓	2	g65,g67,g68,g70,g64,g61,g63,g62,g55,g66,
✓	3	g4,g3,g1,g2,g5,g9,g7,g8,g10,g6,g11,g12,g13
✓	4	g13,g14,g16,g17,g18,g19,g15,g20,g21,g22,
✓	5	g2,g5,g6,g7,g8,g9,g10,g11,g13,g14,g15,g16,
✓	6	g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g1
✓	7	g86,g87,g88,g89,g90,g91,g92,g93,g94,g95,
✓	8	g55,g56,g58,g59,g60,g57,g52,g46,g48,g49,
✓	10	g81,g73,g74,g75,g85,g84,g76,g77,g83,g78,
✓	11	g13,g14,g15,g16,g17,g18,g19,g20,g21,g22,
✓	12	g73,g74,g75,g76,g77,g78,g79,g80,g81,g82,
✓	13	g45,g34,g35,g43,g37,g42,g33,g31,g36,g32,
✓	14	g34,g35,g37,g27,g31,g42,g32,g40,g36,g33,
✓	15	g40,g46,g47,g48,g49,g50,g51,g52,g53,g54,
✓	16	g22,g55,g56,g57,g58,g59,g60,g61,g62,g63,
✓	17	g26,g38,g28,g30,g39
✓	18	g41,g28,g30,g38,g39
✓	19	g28,g38,g39,g40,g30
✓	20	g25,g39,g28,g29
✓	21	g39,g46,g47,g48,g49,g50,g51,g52,g53,g54,
✓	22	g71,g72,g73,g74,g75,g76,g77,g78,g79,g80,
✓	23	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,
✓	24	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,
✓	25	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,

Figure 6. 10 Graph view of a bicluster set.

Graph view: Graph view renders the graphical view of each bicluster as a bipartite graph (Fig 6.10). Rendering window is divided by an equal sized small rectangle. Each rectangle renders a bicluster by placing each side of nodes by oval-shaped arrangement. Red nodes indicate the overlapping nodes. Each node is plotted only once hence some of the regions become blank as all of its nodes are somehow overlapped by other biclusters.

Single view of a bicluster: Double-clicking any region will produce a single view of the corresponding bicluster. Single view extracts all nodes including overlapping and renders a clear view of a bicluster.



D:\Bi cluster compare\alias\Env3\toDpcluso.xlsx

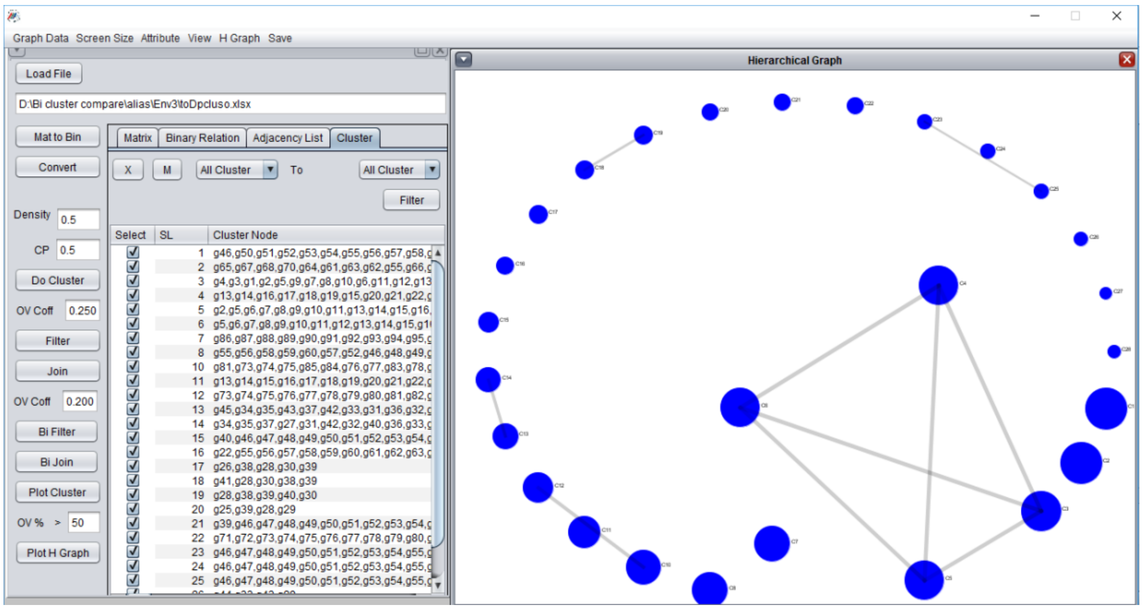
Density 0.5
CP 0.5
OV Coff 0.250
OV Coff 0.200
OV % > 50

```
20  g25,g39,g28,g29
21  g39,g46,g47,g48,g49,g50,g51,g52,g53,g54,g
22  g71,g72,g73,g74,g75,g76,g77,g78,g79,g80,g
23  g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,g
24  g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,g
25  g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,g
```

Figure 6. 11 Magnify a single bicluster.

Matrix view: Matrix view (Fig 6.9) arranges the biclusters by descending order according to their size and plot accordingly.

Hierarchical Graph (for biclustering): In the case of bicluster set, the hierarchical graph is created considering the relation among biclusters in terms of common nodes shared by them (Fig 6.13). We represent such relation as a simple graph with varying nodes size and edge width. Node size for such hierarchical graph is measured by taking the corresponding cluster's total node number in logarithmic scale. Edges width between such nodes is the number of overlapping nodes of the corresponding clusters in logarithmic scale. The parameter in the text field "OV%" is used to calculate the minimum number of the overlapping percentage to be considered while drawing an edge. Edge thickness is proportional to the overlapping percentage between two biclusters. Enter the value as a percentage in the text field and click “Plot H graph”



D:\Bi cluster compare\alias\Env3\toDpcluso.xlsx

Density 0.5

CP 0.5

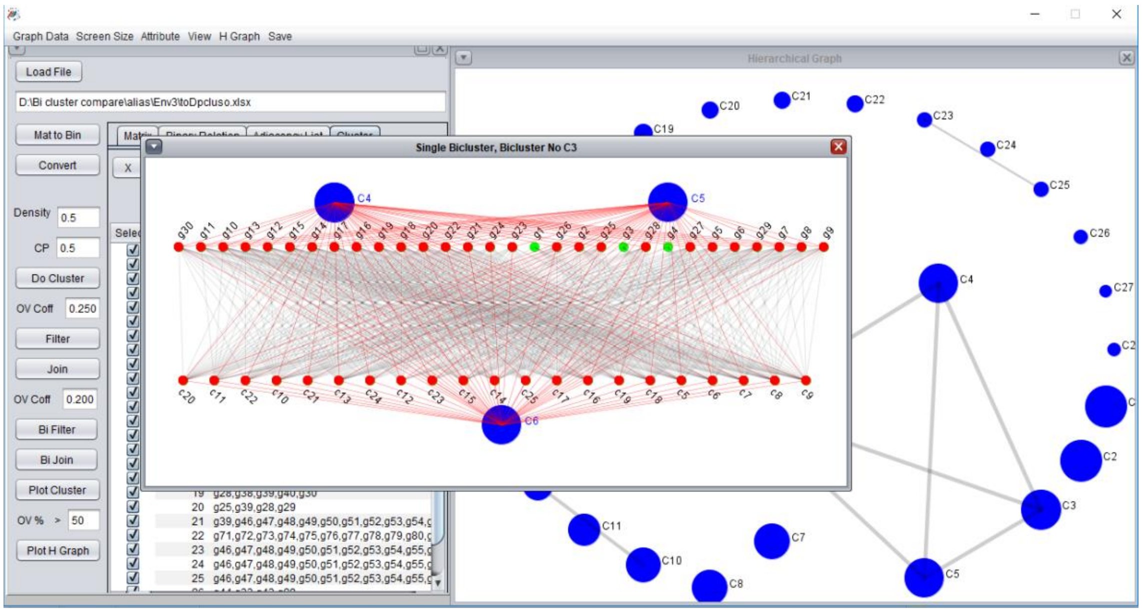
OV Coff 0.250

OV Coff 0.200

OV % > 50

Select	SL	Cluster Node
✓	1	g46,g50,g51,g52,g53,g54,g55,g56,g57,g58,
✓	2	g65,g67,g68,g70,g64,g61,g63,g62,g55,g66,
✓	3	g4,g3,g1,g2,g5,g9,g7,g8,g10,g6,g11,g12,g13
✓	4	g13,g14,g16,g17,g18,g19,g15,g20,g21,g22,
✓	5	g2,g5,g6,g7,g8,g9,g10,g11,g13,g14,g15,g16,
✓	6	g5,g6,g7,g8,g9,g10,g11,g12,g13,g14,g15,g1
✓	7	g86,g87,g88,g89,g90,g91,g92,g93,g94,g95,
✓	8	g55,g56,g58,g59,g60,g57,g52,g46,g48,g49,
✓	10	g81,g73,g74,g75,g85,g84,g76,g77,g83,g78,
✓	11	g13,g14,g15,g16,g17,g18,g19,g20,g21,g22,
✓	12	g73,g74,g75,g76,g77,g78,g79,g80,g81,g82,
✓	13	g45,g34,g35,g43,g37,g42,g33,g31,g36,g32,
✓	14	g34,g35,g37,g27,g31,g42,g32,g40,g36,g33,
✓	15	g40,g46,g47,g48,g49,g50,g51,g52,g53,g54,
✓	16	g22,g55,g56,g57,g58,g59,g60,g61,g62,g63,
✓	17	g26,g38,g28,g30,g39
✓	18	g41,g28,g30,g38,g39
✓	19	g28,g38,g39,g40,g30
✓	20	g25,g39,g28,g29
✓	21	g39,g46,g47,g48,g49,g50,g51,g52,g53,g54,
✓	22	g71,g72,g73,g74,g75,g76,g77,g78,g79,g80,
✓	23	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,
✓	24	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,
✓	25	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,

Figure 6. 12 Hierarchical relationship of biclusters.



D:\Bi cluster compare\alias\Env3\toDpcluso.xlsx

Density 0.5

CP 0.5

OV Coff 0.250

OV Coff 0.200

OV % > 50

Select	SL	Cluster Node
✓	20	g25,g39,g28,g29
✓	21	g39,g46,g47,g48,g49,g50,g51,g52,g53,g54,
✓	22	g71,g72,g73,g74,g75,g76,g77,g78,g79,g80,
✓	23	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,
✓	24	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,
✓	25	g46,g47,g48,g49,g50,g51,g52,g53,g54,g55,

Figure 6. 13 A single node from hierarchical biclusters.

Hierarchical bicluster node: Clicking any node in the hierarchical view a new window will appear expanding the selected node and its adjacent overlapping nodes where red nodes indicate the common nodes between two biclusters.

Save Picture: Menu item save option has different submenu to save the picture of different Bicluster set, a single node of different Bicluster, hierarchical relation of different cluster/ Bicluster set, a single node of the hierarchical graph of Bicluster. In order to save any picture, the user has to highlight the appropriate window by clicking the window header and select the corresponding submenu item from the Save menu.

6.5 Conclusion

we implement the DPClusO and BiClusO algorithms using GUI for finding clusters in simple and bipartite graphs. The main purpose of our effort is to provide free and user-friendly software for network analysis. Two illustrative examples explain the procedure of creating, filtering and joining the clusters. Hierarchical graph explain the relationship of different clusters in terms of overlapping nodes or edges. Also java based open sourced implementation will allow user to customize and run the software in different platfor

Chapter 7

Future work and Conclusions

7.1 Future Work

7.1.1 Graph plotting layout

In order to understand the intricate relationship from a network, it is important to plot the graph by using human interpretable layout. Different layout options allow the user to get the internal relationship of a graph at a glance before applying clustering algorithms. In our system, we only use circular, random and combination of circular and random layouts. A layered layout or layered layout by using the descending order of node degree will allow the user to grasp the priority nodes in terms of relations.

7.1.2 Sensitivity of different parameter

In our algorithm, we used two different parameters i.e. relation number and Tanimoto coefficients. The sensitivity is an important issue on selecting the optimal value of these parameters which will allow the user to extract a minimal bicluster set. In chapter three we discussed the sensitivity by using three different attributes which are cluster coefficient, Node number and the number of elements in each row or column. In the future, we will incorporate different graph related to these parameters in our system to aid the user to select the optimal parameter value.

7.1.3 Reducing the execution time

The execution time of our algorithm and rendering the cluster set and graph are important issues which should be taken care of. In our system, a lot of code segments are independent or loosely coupled where parallel processing can be applied. Most of these segments are reading large excel file, calculating and rendering graph coordinate to large canvass, large matrix manipulation using different row-wise and column-wise operation, calculating different weight, and path of edges and nodes from adjacency list. A significant portion of execution time depends on those code segment. Using parallel processing or saving some data in the appropriate data structure by precalculating the redundant task can improve the performance of our system

7.1.4 Inline hypergeomatric test

Most of the generated clusters, we used third-party tools from R to calculate *p*-values based on the hypergeometric test. Adding an additional attribute for classification can be useful to assess the hypergeometric test within our system. Thus the user can get the result along with the richness of each cluster with a specific class.

7.2 Conclusions

Almost all disciplines of modern science and research are generating a huge amount of data through various experiments and analysis using high throughput equipment including computers. Finding useful information from such data is difficult without prior knowledge of data mining algorithms and any programming languages.

In this research we create a new biclustering algorithm BiClusO, We also compare our algorithm to different well-known biclustering algorithms using two different biological datasets and four different synthetic datasets. Our algorithm shows the best performance which influences us to build a new software for network analysis.

We create the software DPClusSBO using GUI and implementing the BiClusO algorithm and previously developed DPClusO algorithm. DPClusSBO is an easy and user-friendly tool for finding gist information from a simple and bipartite network consisting of many nodes and edges. Any types of data can be transformed into a network by preprocessing and then DPClusSBO can be applied. One of the distinctive features of our software is the overlapping controlling parameter which is used to merge and filter the cohesive group of a cluster set to a certain extent.

We apply our tools for analyzing different datasets from KNApSAck database. Using simple clustering from our tool we successfully identify some structurally similar chemical compound. We apply biclustering to Species–VOC bipartite relationship which identifies similar categorical species and their corresponding VOC. We apply simple clustering to cluster VOC and find some SSVGs which can help the pathway analysis of VOC generation on different species. The relationship of SSVGs produced by different species in terms of kingdom level reveals the conservation of evolutionary hierarchy in terms of biochemical traits.

We also apply the simple clustering for finding IBD related genes to simple PPI network. Our experiment successfully finds out some IBD associated genes with biological significance. We used these genes along with different dataset of IBD genes to analyze the bipartite relationship of miRNA-gene. We apply biclustering to four different data sources of miRNA-gene and mapped those IBD genes to isolate the sub-MRM. Our ranking formula finds some miRNAs where the IBD or related disease associations are revealed in different literature.

Moreover, during our research work, we apply our tool for finding an effective formula from the plant–disease relationship of Unani and ayurvedic data. We also use the tool for analyzing and finding the properties of sulfur contained metabolite emitted from different species.

This software can be used by any engineering, scientific or research field to analyze the relationship of a single feature as a simple graph or duel features as a bipartite graph.

Reference

[1] Pontes, Beatriz, Ra´ul Gir´aldez, and Jes´us S. Aguilar-Ruiz. "Biclustering on expression data: A review." Journal of biomedical informatics 57 (2015): 163-180.

[2] Williams, Andrew, and Sabina Halappanavar. "Application of biclustering of gene expression data and gene set enrichment analysis methods to identify potentially disease causing nanomaterials." Beilstein journal of nanotechnology 6, no. 1 (2015): 2438-2448.

[3] Bebek, Gurkan, and Jiong Yang. "PathFinder: mining signal transduction pathway segments from protein-protein interaction networks." BMC bioinformatics 8, no. 1 (2007): 335.

[4] Henriques, Rui, and Sara C. Madeira. "BicPAM: Pattern-based biclustering for biomedical data analysis." Algorithms for Molecular Biology 9, no. 1 (2014): 27.

[5] Wang, Shu, Robin R. Gutell, and Daniel P. Miranker. "Biclustering as a method for RNA local multiple sequence alignment." Bioinformatics 23, no. 24 (2007): 3289-3296.

[6] de Castro, Pablo AD, Fabr´ıcio O. de Fran¸ca, Hamilton M. Ferreira, and Fernando J. Von Zuben. "Applying biclustering to text mining: an immune-inspired approach." In International Conference on Artificial Immune Systems, pp. 83-94. Springer, Berlin, Heidelberg, 2007.

[7] Banerjee, Arindam, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. "A generalized maximum entropy approach to bregman co-clustering and matrix approximation." Journal of Machine Learning Research 8, no. Aug (2007): 1919-1986

[8] Gnatyshak, Dmitry, Dmitry I. Ignatov, Alexander Semenov, and Jonas Poelmans. "Gaining insight in social networks with biclustering and triclustering." In international conference on business informatics research, pp. 162-171. Springer, Berlin, Heidelberg, 2012.

[9] Huang, Qinghua. "A biclustering technique for mining trading rules in stock markets." In International Conference on Applied Informatics and Communication, pp. 16-24. Springer, Berlin, Heidelberg, 2011.

[10] Muto-Fujita A, Takemoto K, Kanaya S, Nakazato T, Tokimatsu T, Matsumoto N, Kono M, Chubachi Y, Ozaki K and Kotera M, "Data integration aids understanding of butterfly-host plant networks," Sci Rep. 2017 Mar 6;7:43368. doi: 10.1038/srep43368

[11] Altaf-Ul-Amin, Md, Yoko Shinbo, Kenji Mihara, Ken Kurokawa, and Shigehiko Kanaya. "Development and implementation of an algorithm for detection of protein complexes in large interaction networks." BMC bioinformatics 7, no. 1 (2006): 207.

[12] Altaf-Ul-Amin, M Wada, S.Kanaya, "Partitioning a PPI Network into Overlapping Modules Constrained by High-Density and Periphery Tracking" International Scholarly Research Network, vol. 2012, Article ID 726429, 2012.

[13] Altaf-Ul-Amin, Md, et al. "DPClus: a density-periphery based graph clustering software mainly focused on detection of protein complexes in interaction networks." Journal of Computer Aided Chemistry 7 (2006): 150-156..

[14] Cheng Y,Church GM, "Biclustering of expression data," Proc Int Conf Intell Syst Mol Biol. 2000;8:93-103.

[15] Nakamura, Yukiko, et al. "KNApSAcK metabolite activity database for retrieving the relationships between metabolites and biological activities." Plant and Cell Physiology 55.1 (2014): e7-e7.

[16] Afendi, Farit M., Naoaki Ono, Yukiko Nakamura, Kensuke Nakamura, Latifah K. Darusman, Nelson Kibinge, Aki Hirai Morita et al. "Data mining methods for omics and knowledge of crude medicinal plants toward big data biology." Computational and Structural Biotechnology Journal 4, no. 5 (2013): e201301010.

[17] Abdullah, Azian Azamimi, Md Altaf-Ul-Amin, Naoaki Ono, Tetsuo Sato, Tadao Sugiura, Aki Hirai Morita, Tetsuo Katsuragi, Ai Muto, Takaaki

Nishioka, and Shigehiko Kanaya. "Development and mining of a volatile organic compound database." BioMed research international 2015 (2015).

[18] Amos Tanay, Roded Sharan, Ron Shamir; "Discovering statistically significant biclusters in gene expression data," Bioinformatics, Volume 18, Issue suppl 1, 1 July 2002, Pages S136S144.

[19] Sepp Hochreiter, Ulrich Bodenhofer, Martin Heusel, Andreas Mayr, Andreas Mitterecker, Adetayo Kasim, Tatsiana Khamiakova, Suzy Van Sanden, Dan Lin, Willem Talloen, Luc Bijnens, Hinrich W. H. Gohlmann, Ziv Shkedy, Djork-Arn ¨ e Clevert "FABIA: factor ´ analysis for bicluster acquisition," Bioinformatics, Volume 26, Issue 12, 15 June 2010, Pages 15201527

[20] Murali, T. M., and Simon Kasif. "Extracting conserved gene expression motifs from gene expression data." In Biocomputing 2003, pp. 77-88. 2002.

[21] Kluger, Yuval, Ronen Basri, Joseph T. Chang, and Mark Gerstein. "Spectral biclustering of microarray data: coclustering genes and conditions." Genome research 13, no. 4 (2003): 703-716.

[22] Li, Guojun, Qin Ma, Haibao Tang, Andrew H. Paterson, and Ying Xu. "QUBIC: a qualitative biclustering algorithm for analyses of gene expression data." Nucleic acids research 37, no. 15 (2009): e101-e101.

[23] Bergmann, Sven, Jan Ihmels, and Naama Barkai. "Iterative signature algorithm for the analysis of large-scale gene expression data." Physical review E 67, no. 3 (2003): 031902.

[24] Karim Mohammad Bozlul, Huang, Ming, O. N. O. Naoaki, Shigehiko Kanaya, and Md Altaf-Ul-Amin. "BiClusO: A novel biclustering approach and its application to species-VOC relational data." IEEE/ACM transactions on computational biology and bioinformatics(2019)

[25] Lazzeroni, Laura, and Art Owen. "Plaid models for gene expression data." Statistica sinica (2002): 61-86.

[26] Preli, Amela, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter

B¨uhlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. "A systematic comparison and evaluation of biclustering methods for gene expression data." Bioinformatics 22, no. 9 (2006): 1122-1129.

[27] R. Santamar´ıa, R. Ther´on, and L. Quintales, "Bicoverlapper 2.0: visual analysis for gene expression," Bioinformatics, p. btu120, 2014.

[28] M. Streit, S. Gratzl, M. Gillhofer, A. Mayr, A. Mitterecker, and S. Hochreiter, "Furby: fuzzy force-directed bicluster visualization," BMC bioinformatics, vol. 15, no. Suppl 6, p. S4, 2014

[29] J. Heinrich, R. Seifert, M. Burch, and D. Weiskopf, "Bicluster viewer: a visualization tool for analyzing gene expression data," in Advances in Visual Computing, pp. 641–652, Springer, 2011.

[30] J. P. Gon¸calves, S. C. Madeira, and A. L. Oliveira, "Biggests: integrated environment for biclustering analysis of time series gene expression data," BMC research notes, vol. 2, no. 1, p. 124, 2009.

[31] Eren, Kemal, Mehmet Deveci, Onur K¨u¸c¨uktun¸c, and Umit V. C¸ ataly¨urek. "A comparative analysis of ¨ biclustering algorithms for gene expression data." Briefings in bioinformatics 14, no. 3 (2012): 279-292.

[32] Li, Li, Yang Guo, Wenwu Wu, Youyi Shi, Jian Cheng, and Shiheng Tao. "A comparison and evaluation of five biclustering algorithms by quantifying goodness of biclusters for gene expression data." BioData mining 5, no. 1 (2012): 8

[33] Sebastian Kaiser, Rodrigo Santamaria, Tatsiana Khamiakova, Martin Sill, Roberto Theron, Luis Quintales, Friedrich Leisch and Ewoud De Troyer. "Package biclust", Title BiCluster Algorithms ,Version 2.0.1, Date 2018-06-09

[34] Mohammad Bozlul Karim, Naoaki Ono, Md.Altaf-Ul-Amin and Shigehiko Kanaya. "Classification of species by biclustering based on emitting volatile organic compounds." APBC 2018 conference. Yokohama, Japan, 15-17

[35] Brown, Michael PS, William Noble Grundy, David Lin, Nello Cristianini, Charles Walsh Sugnet, Terrence S. Furey, Manuel Ares, and David Haussler. "Knowledge-based analysis of microarray gene expression data by using

support vector machines." Proceedings of the National Academy of Sciences97, no. 1 (2000): 262-267.

[36] Mateos, Alvaro, Joaquin Dopazo, Ronald Jansen, Yuhai Tu, Mark Gerstein, and Gustavo Stolovitzky. "Systematic learning of gene functional classes from DNA array expression data by using multilayer perceptrons." Genome Research 12, no. 11 (2002): 1703-1715

[37] Lægreid, Astrid, Torgeir R. Hvidsten, Herman Midelfart, Jan Komorowski, and Arne K. Sandvik. "Predicting gene ontology biological process from temporal gene expression patterns." Genome research 13, no. 5 (2003): 965-979.

[38] Raghava, Gajendra PS, and Joon H. Han. "Correlation and prediction of gene expression level from amino acid and dipeptide composition of its protein." BMC bioinformatics 6, no. 1 (2005): 59.

[39] Altaf-Ul-Amin, Md, Tetsuo Katsuragi, Tetsuo Sato, Naoaki Ono, and Shigehiko Kanaya. "An unsupervised approach to predict functional relations between genes based on expression data." BioMed research international 2014 (2014).

[40] Eguchi, R., Karim, M. B., Hu, P., Sato, T., Ono, N., Kanaya, S., and Altaf-Ul-Amin, M. (2018). "An integrative network-based approach to identify novel disease genes and pathways: a case study in the context of inflammatory bowel disease." BMC Bioinformatics, 19(1), 264.

[41] Hossain, Shaikh Farhad, Sony Hartono Wijaya, Ming Huang, Irmanida Batubara, Shigehiko Kanaya, and Md Altaf-Ul-Amin Farhad. "Prediction of Plant-Disease Relations Based on Unani Formulas by Network Analysis." In 2018 IEEE 18th International Conference on Bioinformatics and Bioengineering (BIBE), pp. 348-351. IEEE, 2018

[42] Uitert, Miranda van, Wouter Meuleman, and Lodewyk Wessels. "Biclustering sparse binary genomic data." Journal of Computational Biology 15, no. 10 (2008): 1329-1345.

[43] Dudareva, Natalia, et al. "Biosynthesis, function and metabolic engineering of plant volatile organic compounds." New Phytologist 198.1 (2013): 16-32.

[44] Hatipoglu, Seda Damla, et al. "Determination of volatile organic compounds in fourty five Salvia species by thermal desorption-GCMS technique." Records of Natural Products10.6 (2016): 659.

[45] Lee, Samantha, et al. "Volatile organic compounds emitted by Trichoderma species mediate plant growth." Fungal biology and biotechnolo-gy 3.1 (2016): 7.

[46] Hinge, Vidya R., Hemant B. Patil, and Altafhusain B. Nadaf. "Aroma volatile analyses and 2AP characterization at various developmental stages in Basmati and Non-Basmati scented rice (Oryza sativa L.) cultivars." Rice 9.1 (2016): 38.

[47] Ascrizzi, Roberta, et al. "Patterns in Volatile Emission of Different Aerial Parts of Caper (Capparis spinosa L.)." Chemistry and biodiversity 13.7 (2016): 904-912.

[48] Kong, Ying, et al. "Floral scents produced by Lilium and Cardiocrinum species native to China." Biochemical Systematics and Ecology 70 (2017): 222-229.

[49] Wiebelhaus, Nancy, Natasha M. Kreitals, and Jose R. Almirall. "Differentiation of marijuana headspace volatiles from other plants and hemp products using capillary microextraction of
volatiles (CMV) coupled to gas-chromatographymass spectrometry (GCMS)." Forensic Chemistry 2 (2016): 1-8.

[50] Kusano, Miyako, et al. "Unbiased profiling of volatile organic compounds in the headspace of Allium plants using an in-tube extraction device." BMC research notes 9.1 (2016): 133.

[51] Yamani, Hanaa A., et al. "Antimicrobial activity of Tulsi (Ocimum tenuiflorum) essential oil and their major constituents against three species of bacteria." Frontiers in microbiology 7 (2016): 681.

[52] Hoe, Yin Chen, et al. "Flowering mechanisms, pollination strategies and floral scent analyses of syntopically coflowering Homalomena spp.(Araceae) on Borneo." Plant Biology 18.4 (2016): 563- 576.

[53] Zhigzhitzhapova, Svetlana Vasylievna, et al. "Chemical composition of volatile organic compounds of Artemisia vulgaris L.(Asteraceae) from the QinghaiTibet Plateau." Industrial Crops and Products 83 (2016): 462-469.

[54] Domik, Dajana, et al. "A Terpene synthase is involved in the synthesis of the volatile organic compound sodorifen of Serratia plymuthica 4Rx13." Frontiers in microbiology 7 (2016): 737.

[55] Erbas, Sabri, and Hasan Baydar. "Variation in scent compounds of oil-bearing rose (Rosa damascena Mill.) produced by headspace solid phase microextraction, hydrodistillation and solvent extraction." Records of Natural Products 10.5 (2016): 555.

[56] Santos, Andrili Borges, et al. "Biogeneration of volatile organic compounds produced by Phormidium autumnale in heterotrophic bioreactor." Journal of applied phycology 28.3 (2016): 1561-1570.

[57] Torbati, S., A. Movafeghi, and Dj Djozan. "Identification of Volatile Organic Compounds Released from the Leaves and Flowers of Artemisia austriaca Using the Modified Pencil Lead as a Fibre of Solid Phase Microextraction." Journal of Essential Oil Bearing Plants 19.5 (2016): 1224-1233.

[58] Karmakar, Amarnath, Abhishek Mukherjee, and Anandamay Barik. "Floral volatiles with colour cues from two cucurbitaceous plants causing attraction of Aulacophora foveicollis." Entomologia Experimentalis et Applicata 158.2 (2016): 133-141.

[59] Cozzolino, Rosaria, et al. "Determination of volatile organic compounds in the dried leaves of Salvia species by solid-phase microextraction coupled to gas chromatography mass spectrometry." Natural product research 30.7 (2016): 841-848.

[60] De Vrieze, Mout, et al. "Volatile organic compounds from native potato-associated Pseudomonas as potential anti-oomycete agents." Frontiers in microbiology 6 (2015): 1295.

[61] Killiny, Nabil, and Shelley E. Jones. "Profiling of volatile organic compounds released from individual intact juvenile and mature citrus leaves." Journal of plant physiology 208 (2017): 47-51.

[62] Jaeger, Deidre M., Justin B. Runyon, and Bryce A. Richardson. "Signals of speciation: volatile organic compounds resolve closely related sage brush taxa, suggesting their importance in evolution." New Phytologist211.4 (2016): 1393-1401.

[63] Liarzi, Orna, et al. "Use of the endophytic fungus Daldinia cf. concentrica and its volatiles as bio-control agents." PloS one11.12 (2016): e0168242.

[64] Groenhagen, Ulrike, et al. "Coupled biosynthesis of volatiles and salino-sporamide A in Salinispora tropica." ChemBioChem17.20 (2016): 1978-1985.

[65] Li, Yuying, et al. "Volatile organic compounds emissions from Luculia pinceana flower and its changes at different stages of flower development." Molecules 21.4 (2016): 531.

[66] Mesquita, Paulo RR, et al. "Discrimination of Eugenia uniflora L. bio-types based on volatile compounds in leaves using HSSPME/GCMS and chemometric analysis." Microchemical Journal 130 (2017): 79-87.

[67] SanchezOrtiz, B. L., et al. "Antifungal, antioomycete and phytotoxic effects of volatile organic compounds from the endophytic fungus Xylaria sp. strain PB3f3 isolated from Haematoxylon
brasiletto." Journal of applied microbiology120.5 (2016): 1313-1325.

[68] Estrella-Parra, Edgar A., et al. "Volatile organic compounds from Pachyrhizus ferrugineus and Pachyrhizus erosus (Fabaceae) leaves." Boletn Latinoamericano y del Caribe de Plantas Medicinales y Aromaticas 15.3 (2016).

[69] Thongpoon, C., and T. Machan. "Aroma Volatile Composition of Millingtonia hortensis Linn. F. Flower Growing in Chiang Rai, Thailand." Asian Journal of Chemistry 28.2 (2016): 329.

[70] Neerincx, A. H., et al. "Identification of Pseudomonas aeruginosa and Aspergillus fumigatus mono and co-cultures based on volatile biomarker combinations." Journal of breath research 10.1 (2016): 016002.

[71] UlloaBen´ıtez, ., et al. "Phytotoxic and antimicrobial activity of volatile and

semivolatile organic compounds from the endophyte Hypoxylon anthochroum strain Blaci isolated from Bursera lancifolia (Burseraceae)." Journal of applied microbiology 121.2 (2016): 380-400

[72] Cuevas, F. J., et al. "Effect of management (organic vs conventional) on volatile profiles of six plum cultivars (Prunus salicina Lindl.). A chemometric approach for varietal classification and determination of potential markers." Food chemistry 199 (2016): 479-484.

[73] Kupska, Magdalena, and Henryk H. Jele. "Intube extraction for the determination of the main volatile compounds in Physalis peruviana L." Journal of separation science 40.2 (2017): 532-541.

[74] Raza, Waseem, et al. "Response of tomato wilt pathogen Ralstonia solanacearum to the volatile organic compounds produced by a biocontrol strain Bacillus amyloliquefaciens SQR-9." Scientific reports 6 (2016): 24856.

[75] Karim, Mohammad Bozlul, Nobutaka Wakamatsu, and Md AltafUl-Amin. "[Dedicated to Prof. T. Okada and Prof. T. Nishioka: data science in chemistry] DPClusOST: A Software Tool for General Purpose Graph Clustering." Journal of Computer Aided Chemistry 18 (2017): 76-93.

[76] Eguchi, R., Karim, M. B., Hu, P., Sato, T., Ono, N., Kanaya, S., and Altaf-Ul-Amin, M. (2018). "An integrative network-based approach to identify novel disease genes and pathways: a case study in the context of inflammatory bowel disease." BMC Bioinformatics, 19(1), 264.

[77] Cai, Yu-Dong, et al. "Prediction of compounds biological function (metabolic pathways) based on functional group composition." Molecular diversity 12.2 (2008): 131-137.

[78] Hamdalla, Mai A., et al. "Metabolic pathway predictions for metabolomics: a molecular structure matching approach." Journal of chemical information and modeling 55.3 (2015): 709-718.

[79] Sanz, Carlos, and Ana G. Prez. "Plant metabolic pathways and flavor biosynthesis." Handbook of fruit and vegetable flavors. Hoboken, NJ, USA: John Wiley & Sons, Inc (2010): 129-55.

[80] Cao, Yiqun, Anna Charisi, Li-Chang Cheng, Tao Jiang, and Thomas Girke. "ChemmineR: a compound mining framework for R." Bioinformatics 24, no. 15 (2008): 1733-1734.

[81] Dunkel M, Gnther S, Ahmed J, Wittig B, Preissner R. SuperPred: "drug classification and target prediction," Nucleic Acids Research. 2008;36(Web Server issue):W55-W59. doi:10.1093/nar/gkn307.

[82] Matter H1,"Selecting optimally diverse compounds from structure databases: a validation study of two-dimensional and threedimensional molecular descriptors," J Med Chem. 1997 Apr 11;40(8):1219-29.

[83] Pennisi, Elizabeth. "Modernizing the tree of life." (2003): 1692- 1697.

[84] Gasparetto, Marco, and Graziella Guariso. "Highlights in IBD epidemiology and its natural history in the paediatric age." Gastroenterology research and practice 2013 (2013).

[85] Walsh CJ, Hu P, Batt J, dos Santos CC. "Discovering MicroRNA-Regulatory Modules in Multi-Dimensional Cancer Genomic Data: A Survey of Computational Methods". Cancer Informatics. 2016; pp.68-73.

[86] Haberman Y, Tickle TL, Dexheimer PJ, Kim M-O, Tang D, Karns R, Baldassano RN, Noe JD, Rosh J, Markowitz J, et al. "Pediatric crohn disease patients exhibit specific ileal transcriptome and microbiome signature". J Clin Investig. 2014;124(8):3617.

[87] Piñero J, Queralt-Rosinach N, Bravo À, Deu-Pons J, Bauer-Mehren A, Baron M, Sanz F, Furlong LI. "DisGeNET: a discovery platform for the dynamical exploration of human diseases and their genes,". Database: J Biol Databases Curation. 2015;2015:028. https://doi.org/10.1093/ database/bav028

[88] Schaefer MH, Fontaine J-F, Vinayagam A, Porras P, Wanker EE, Andrade-Navarro MA. "HIPPIE: Integrating Protein Interaction Networks with Experiment Based Quality Scores". PLoS ONE. 2012;7(2):31826. https://doi.org/10.1371/journal.pone.0031826.

[89] Davis AP, Grondin CJ, Johnson RJ, Sciaky D, King BL, Mcmorran R, Wiegers J, Wiegers TC, Mattingly CJ. "The Comparative Toxicogenomics Database:

update 2017". Nucleic Acids Res. 2017;45:. https://doi.org/10.1093/nar/gkw838.

[90] Yu W, Gwinn M, Clyne M, Yesupriya A, Khoury MJ. "A navigator for human genome epidemiology". Nat Genet. 2008;40(2):124–5. https://doi.org/10.1038/ng0208-124.

[91] Liu JZ, van Sommeren S, Huang H, Ng SC, Alberts, et al. "Association analyses identify 38 susceptibility loci for inflammatory bowel disease and highlight shared genetic risk across populations". Nat Genet. 2015;47(9):979–86. https://doi.org/10.1038/ng.3359.

[92] Anderson CA, Boucher G, Lees CW, Franke, et al. "Meta-analysis identifies 29 additional ulcerative colitis risk loci, increasing the number of confirmed associations to 47". Nat Genet. 2011;43(3):246–52. https://doi.org/10.1038/ng.764

[93] Franke A, McGovern DPB, Barrett JC, Wang, et al. "Genome-wide meta-analysis increases to 71 the number of confirmed Crohn's disease susceptibility loci". Nat Genet. 2010;42(12):1118–25. https://doi.org/10.1038/ng.717.

[94] Barrett JC, Hansoul S, Nicolae DL, Cho JH, Duerr RH, Rioux, et al. "Genome-wide association defines more than 30 distinct susceptibility loci for Crohn's disease". Nat Genet. 2008;4 0(8):
955–62. https://doi.org/10. 1038/ng.175.

[95] Benjamini Y, Hochberg Y. "Controlling the false discovery rate: a practical and powerful approach to multiple testing". J R Stat Soc Ser B Methodol. 1995;57(1):289–300

[96] Glas J, Seiderer J, Wagner J, Olszak T, Fries C, Tillack C, Friedrich M, Beigel F, Stallhofer J, Steib C, et al. "Analysis of il12b gene variants in inflammatory bowel disease". PloS one. 2012;7(3):e34349.

[97] Ye X, Liu S, Hu M, Song Y, Huang H, Zhong Y. "Ccr5 expression in inflammatory bowel disease and its correlation with inflammatory cells and β-arrestin2 expression". Scand J Gastroenterol. 2017;52(5):551–7.

[98] Mora-Buch R, Dotti I, Planell N, Calderón-Gómez E, Jung P, Masamunt M, Llach J, Ricart E, Batlle E, Panés J, et al. "Epithelial il-1r2 acts as a homeostatic regulator during remission of ulcerative colitis". Mucosal Immunol. 2016;9(4):950.

[99] Mohamadzadeh M, Pfeiler EA, Brown JB, Zadeh M, Gramarossa M, Managlia E, Bere P, Sarraj B, Khan MW, Pakanati KC, et al. "Regulation of induced colonic inflammation by lactobacillus acidophilus deficient in lipoteichoic acid". Proc Natl Acad Sci. 2011;108 (Supplement 1):4623–30.

[100] Jin D-Y, Jeang K-T. "Isolation of full-length cdna and chromosomal localization of human nf-κb modulator NEMO to Xq28". J Biomed Sci. 1999;6(2):115–20.

[101] Uhlig HH, Schwerd T, Koletzko S, Shah N, Kammermeier J, Elkadri A, Ouahed J, Wilson DC, Travis SP, Turner D, et al. "The diagnostic approach to monogenic very early onset inflammatory bowel disease". Gastroenterology. 2014;147(5):990–1007.

[102] Andreoletti G, Shakhnovich V, Christenson K, Coelho T, Haggarty R, Afzal NA, Batra A, Petersen B-S, Mort M, Beattie RM, et al. "Exome analysis of rare and common variants within the NOD signaling pathway". Sci Rep. 2017;7:46454.

[103] Borthakur A, Bhattacharyya S, Alrefai WA, Tobacman JK, Ramaswamy K, Dudeja PK. "Platelet-activating factor-induced nf-κb activation and il-8 production in intestinal epithelial cells are bcl10-dependent". Inflamm Bowel Dis. 2009;16(4):593–603

[104] Yu Q, Zhang S, Chao K, Feng R, Wang H, Li M, Chen B, He Y, Zeng Z, Chen M. "E3 ubiquitin ligase RNF183 is a novel regulator in inflammatory bowel disease". J Crohn's Colitis. 2016;10(6):713–25

[105] Schweickart VL, Epp A, Raport CJ, Gray PW. "CCR11 is a functional receptor for the monocyte chemoattractant protein family of chemokines". J Biol Chem. 2000;275(13):9550–6

[106] Wells T, Proudfoot A. "Chemokine receptors and their antagonists in allergic lung disease". Inflamm Res. 1999;48(7):353–62.

[107] Nilsson J, Schoser B, Laforet P, Kalev O, Lindberg C, Romero NB, Dávila López M, Akman HO, Wahbi K, Iglseder S, et al. "Polyglucosan body myopathy caused by defective ubiquitin ligase RBCK1". Ann Neurol. 2013;74(6):914–9.

[108] Knecht C, Fretter C, Rosenstiel P, Krawczak M, Hütt M-T. "Distinct metabolic network states manifest in the gene expression profiles of pediatric inflammatory bowel disease patients and controls". Sci Rep. 2016;6:32584

[109] Drev D, Bileck A, Erdem ZN, Mohr T, Timelthaler G, Beer A, Gerner C Marian B. "Proteomic profiling identifies markers for inflammation-related tumor–fibroblast interaction". Clin Proteomics. 2017;14(1):33

[110] Häcker H, Tseng P-H, Karin M. "Expanding traf function: TRAF3 as a tri-faced immune regulator". Nat Rev Immunol. 2011;11(7):457.

[111] Qiao YQ, Shen J, Gu Y, Tong JL, Xu XT, Huang ML, Ran ZH. "Gene expression of tumor necrosis factor receptor associated-factor (traf)-1 and traf-2 in inflammatory bowel disease". J Dig Dis. 2013;14(5):244–50.

[112] Shen J, Qiao Y, Ran Z, Wang T, Xu J, Feng J. "Intestinal protein expression profile identifies inflammatory bowel disease and predicts relapse". Int J Clin Exp Pathol. 2013;6(5):917.

[113] Shen J, Qiao Y-q, Ran Z-h, Wang T-r. "Up-regulation and pre-activation of traf3 and traf5 in inflammatory bowel disease". Int J Med Sci. 2013;10(2):156.

[114] Arch RH, Gedrich RW, Thompson CB. "Tumor necrosis factor receptor-associated factors (TRAFs)—a family of adapter proteins that regulates life and death". Genes Dev. 1998;12(18):2821–30. https://doi.org/10.1101/gad.12.18.2821.

[115] Ringel-Scaia VM, McDaniel DK, Allen IC. "The goldilocks conundrum: Nlr inflammasome modulation of gastrointestinal inflammation during inflammatory bowel disease". Crit Rev™ Immunol. 2016;36(4).

[116] Zhao H, Jaffer T, Eguchi S, Wang Z, Linkermann A, Ma D. "Role of

necroptosis in the pathogenesis of solid organ injury". Cell Death Dis. 2015;6(11):e1975.

[117] Chen J, Xu H, Aronow BJ, Jegga AG. "Improved human disease candidate gene prioritization using mouse phenotype". BMC Bioinformatics. 2007;8(1):392.

[118] Adie EA, Adams RR, Evans KL, Porteous DJ, Pickard BS. "Suspects: enabling fast and effective prioritization of positional candidates". Bioinformatics. 2006;22(6):773–4.

[119] Tiffin N, Kelso JF, Powell AR, Pan H, Bajic VB, Hide WA. "Integration of text-and data-mining using ontologies successfully selects disease gene" candidates. Nucleic Acids Res. 2005;33(5):1544–52.

[120] Aerts S, Lambrechts D, Maity S, Van Loo P, Coessens B, De Smet F, Tranchevent L-C, De Moor B, Marynen P, Hassan B, et al. "Gene prioritization through genomic data fusion. Nat Biotechnol". 2006;24(5):537

[121] Chen J, Bardes EE, Aronow BJ, Jegga AG. "Toppgene suite for gene list enrichment analysis and candidate gene prioritization". Nucleic Acids Res. 2009;37(suppl_2):305–11.

[122] Sticht C, De La Torre C, Parveen A, Gretz N.: "miRWalk: An online resource for prediction of microRNA binding sites". PLoS One. 2018 Oct 18;13(10):

[123] Vlachos, Ioannis S., et al. "DIANA miRPath v. 2.0: investigating the combinatorial effect of microRNAs in pathways." Nucleic acids research 40.W1 (2012): W498-W504.

[124] Xiao F, Zuo Z, Cai G, Kang S, Gao X, Li T: "miRecords: an integrated resource for microRNA-target interactions". Nucleic Acids Res. 2009, 37: D105-D110.

[125] Chou, Chih-Hung, Sirjana Shrestha, Chi-Dung Yang, Nai-Wen Chang, Yu-Ling Lin, Kuang-Wen Liao, Wei-Chi Huang et al. "miRTarBase update 2018: a resource for experimentally validated microRNA-target interactions." Nucleic acids research 46, no. D1 (2017): D296-D302.

[126] Karim, Mohammad Bozlul, Shigehiko Kanaya, and Md Altaf-Ul Amin. "Comparison of BiClusO with Five Different Biclustering Algorithms Using

Biological and Synthetic Data." International Conference on Complex Networks and their Applications. Springer, Cham, 2018.

[127] Akao, Yukihiro, Yoshihito Nakagawa, and Tomoki Naoe. "let-7 microRNA functions as a potential growth suppressor in human colon cancer cells." Biological and Pharmaceutical Bulletin 29, no. 5 (2006): 903-906.

[128] Zhang, Hong-He, Xian-Jun Wang, Guo-Xiong Li, En Yang, and Ning-Min Yang. "Detection of let-7a microRNA by real-time PCR in gastric carcinoma." World journal of gastroenterology: WJG 13, no. 20 (2007): 2883.

[129] Rutter, Matthew D., Brian P. Saunders, Kay H. Wilkinson, Steve Rumbles, Gillian Schofield, Michael A. Kamm, Christopher B. Williams, Ashley B. Price, Ian C. Talbot, and Alastair Forbes. "Thirty-year analysis of a colonoscopic surveillance program for neoplasia in ulcerative colitis." Gastroenterology 130, no. 4 (2006): 1030-1038.

[130] Eaden, J. A., K. R. Abrams, and J. F. Mayberry. "The risk of colorectal cancer in ulcerative colitis: a meta-analysis." Gut 48, no. 4 (2001): 526-535.

[131] Ullman, Thomas A., and Steven H. Itzkowitz. "Intestinal inflammation and cancer." Gastroenterology 140, no. 6 (2011): 1807-1816.

[132] Ryan W. Stidham, Peter D.R. Higgins. "Colorectal Cancer in Inflammatory Bowel Disease" Clinics in Colon and Rectal Surgery 2018; 31(03): 168-178 DOI: 10.1055/s-0037-1602237.

[133] Berindan-Neagoe, Ioana, and George A. Calin. "Molecular pathways: microRNAs, cancer cells, and microenvironment." Clinical Cancer Research 20, no. 24 (2014): 6247-6253.

[134] Viswanathan, Srinivas R., and George Q. Daley. "Lin28: A microRNA regulator with a macro role." Cell 140, no. 4 (2010): 445-449.

[135] King, C. E., Louise Wang, Rafael Winograd, B. B. Madison, P. S. Mongroo, C. N. Johnstone, and Anil K. Rustgi. "LIN28B fosters colon cancer migration, invasion and transformation through let-7-dependent and-independent mechanisms." Oncogene 30, no. 40 (2011): 4185.

[136] Tu, Ho-Chou, Sarah Schwitalla, Zhirong Qian, Grace S. LaPier, Alena

Yermalovich, Yuan-Chieh Ku, Shann-Ching Chen et al. "LIN28 cooperates with WNT signaling to drive invasive intestinal and colorectal adenocarcinoma in mice and humans." Genes & development 29, no. 10 (2015): 1074-1086.

[137] Wang, Tianzhen, Guangyu Wang, Dapeng Hao, Xi Liu, Dong Wang, Ning Ning, and Xiaobo Li. "Aberrant regulation of the LIN28A/LIN28B and let-7 loop in human malignant tumors and its effects on the hallmarks of cancer." Molecular cancer 14, no. 1 (2015): 125.

[138] Chen, Wenfeng, Guosheng Lin, Yizhou Yao, Jishen Chen, Hanli Shui, Qinghai Yang, Xiaoya Wang et al. "MicroRNA hsa-let-7e-5p as a potential prognosis marker for rectal carcinoma with liver metastases." Oncology letters 15, no. 5 (2018): 6913-6924.

[139] Guo, Zhen, Rong Wu, Jianfeng Gong, Weiming Zhu, Yi Li, Zhiming Wang, Ning Li, and Jieshou Li. "Altered micro RNA expression in inflamed and non- inflamed terminal ileal mucosa of adult patients with active C rohn's disease." Journal of gastroenterology and hepatology 30, no. 1 (2015): 109-116.

[140] Kolenda, Tomasz, Weronika Przybyła, Anna Teresiak, Andrzej Mackiewicz, and Katarzyna M. Lamperska. "The mystery of let-7d–a small RNA with great power." Contemporary oncology18, no. 5 (2014): 293.

[141] Zhang, Jia-Xing, Wu Song, Zhen-Hua Chen, Jin-Huan Wei, Yi-Ji Liao, Jian Lei, Ming Hu et al. "Prognostic and predictive value of a microRNA signature in stage II colon cancer: a microRNA expression analysis." The lancet oncology 14, no. 13 (2013): 1295-1306.

[142] Wu, Feng, Simin Zhang, Themistocles Dassopoulos, Mary L. Harris, Theodore M. Bayless, Stephen J. Meltzer, Steven R. Brant, and John H. Kwon. "Identification of microRNAs associated with ileal and colonic Crohn's disease." Inflammatory bowel diseases 16, no. 10 (2010): 1729-1738.

[143] Ni, Shujuan, Weiwei Weng, Midie Xu, Qifeng Wang, Cong Tan, Hui Sun, Lei Wang, Dan Huang, Xiang Du, and Weiqi Sheng. "miR-106b-5p inhibits the invasion and metastasis of colorectal cancer by targeting CTSA."

OncoTargets and therapy 11 (2018): 3835.

[144] Honardoost, Mohammad Amin, Reza Naghavian, Fereshteh Ahmadinejad, Aref Hosseini, and Kamran Ghaedi. "Integrative computational mRNA–miRNA interaction analyses of the autoimmune-deregulated miRNAs and well-known Th17 differentiation regulators: An attempt to discover new potential miRNAs involved in Th17 differentiation." Gene 572, no. 2 (2015): 153-162.

[145] Lü, M. H., B. Tang, S. Zeng, C. J. Hu, R. Xie, Y. Y. Wu, S. M. Wang, F. T. He, and S. M. Yang. "Long noncoding RNA BC032469, a novel competing endogenous RNA, upregulates hTERT expression by sponging miR-1207-5p and promotes proliferation in gastric cancer." Oncogene 35, no. 27 (2016): 3524.

[146] Zhao, Ci, Qi Zhao, Chunhui Zhang, Guangyu Wang, Yuanfei Yao, Xiaoyi Huang, Fei Zhan et al. "miR-15b-5p resensitizes colon cancer cells to 5-fluorouracil by promoting apoptosis via the NF-κB/XIAP axis." Scientific reports 7, no. 1 (2017): 4194.

[147] Yue, Ben, Bo Sun, Chenchen Liu, Senlin Zhao, Dongyuan Zhang, Fudong Yu, and Dongwang Yan. "Long non- coding RNA Fer- 1- like protein 4 suppresses oncogenesis and exhibits prognostic value by associating with miR- 106a- 5p in colon cancer." Cancer science 106, no. 10 (2015): 1323-1332.

[148] Omidbakhsh, Ameneh et al. "Micro-RNAs -106a and -362-3p in Peripheral Blood of Inflammatory Bowel Disease Patients." The open biochemistry journalvol. 12 78-86. 29 Jun. 2018, doi:10.2174/1874091X01812010078

[149] Koukos, Georgios, Christos Polytarchou, Jess L. Kaplan, Alessio Morley–Fletcher, Beatriz Gras–Miralles, Efi Kokkotou, Mariah Baril–Dore, Charalabos Pothoulakis, Harland S. Winter, and Dimitrios Iliopoulos. "MicroRNA-124 regulates STAT3 expression and is down-regulated in colon tissues of pediatric patients with ulcerative colitis." Gastroenterology145, no. 4 (2013): 842-852.

[150] Deng, Danni, Lei Wang, Yao Chen, Bowen Li, Lian Xue, Naiyuan Shao, Qiang Wang, Xiwei Xia, Yilin Yang, and Feng Zhi. "MicroRNA- 124- 3p regulates

cell proliferation, invasion, apoptosis, and bioenergetics by targeting PIM1 in astrocytoma." Cancer science 107, no. 7 (2016): 899-907.

[151] Lee, Juneyoung, Eun Jeong Park, and Hiroshi Kiyono. "MicroRNA-orchestrated pathophysiologic control in gut homeostasis and inflammation." BMB reports 49, no. 5 (2016): 263.

[152] Jia, Litao, Jiamin Chen, Chuangao Xie, Liming Shao, Zhipeng Xu, and Lu Zhang. "microRNA-1228* impairs the pro-angiogenic activity of gastric cancer cells by targeting macrophage migration inhibitory factor." Life sciences 180 (2017): 9-16.

[153] Fan, Yannan, and Jianguo Xia. "miRNet—Functional Analysis and Visual Exploration of miRNA–Target Interactions in a Network Context." In Computational Cell Biology, pp. 215-233. Humana Press, New York, NY, 2018.

[154] E. Georgii, S. Dietmann, T. Uno, P. Pagel, and K. Tsuda, "Enumeration of condition-dependent dense modules in protein interaction networks," Bioinformatics, vol. 25, no. 7, pp. 933–940, 2009.

[155] Sheridan, R.P. et al., 2016. "Extreme Gradient Boosting as a Method for Quantitative Structure-Activity Relationships". Journal of Chemical Information and Modeling

[156] Junshui Ma et al. "Deep Neural Nets as a Method for Quantitative Structure−Activity Relationships", J. Chem. Inf. Model., 2015, 55 (2), pp 263–274 DOI: 10.1021/ci500747n

Achievements

Peer review journal paper

Mohammad Bozlul Karim , Nobutaka Wakamatsu,Md. Altaf-Ul-Amin "DPClusOST: A software tool for general purpose graph clustering". ,"Journal of Computer Aided Chemistry 18(0), 76-93, 2017

Mohammad Bozlul Karim, Ming Huang, Naoaki ONO, Shigehiko Kanaya, Md. Altaf-Ul Amin ; "BiClusO: A novel biclustering approach and its application to species-VOC relationship data " IEEE TRANSACTIONS ON Computational Biology and Bioinformatics 2019.

Mohammad Bozlul Karim, Shigehiko Kanaya, Md. Altaf-Ul-Amin; Implementation of BiClusO and its comparison with other biclustering algorithms; Applied Network Science (Springer)

Peer review journal paper (Submitted, Under review):

Mohammad Bozlul Karim, Shigehiko Kanaya, Md. Altaf-Ul-Amin; DPClusSBO: An integrated software for clustering of simple and bipartite graphs. SoftwareX (ELSEVIER);

Peer review international conference

Mohammand Bozlul Karim; Md. Altaf-Ul-Amin; Shigehiko Kanaya; "Classification of species by bi-clustering based on emitting volatile organic compounds". Asia Pacific Bioinformatics Conference (APBC) 2018

Mohammand Bozlul Karim; Naoaki ONO; Pingzhao Hu ; Shigehiko Kanaya;Md. Altaf-Ul-Amin; “Identification of IBD associated miRNAs by bi - clustering” International Conference on Computational Mathematics , Physics and It's Applications (ICCMPA 2018)

Bozlul Karim, Mohammad & Kanaya, Shigehiko & Amin, Altaf. (2019). Comparison of BiClusO with Five Different Biclustering Algorithms Using Biological and Synthetic Data: Volume 2 Proceedings The 7th International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2018. 10.1007/978-3-030-05414-4_46.

Peer review international conference (Submitted)

Md. Altaf-Ul-Amin; Mohammand Bozlul Karim; Pingzhao Hu; Naoaki ONO; Shigehiko Kanaya; “Discovery of Inflammatory Bowel Disease-associated miRNAs using a Novel Bipartite Clustering Approach” The 30th International Conference on Genome Informatics (GIW 2019)

Appendices

Table 1 : The result of biclustering with species name and volatile organic compound PubChem ID. CID= Cluster ID Taxonomy expressed mostly in family level indicates the maximum number of similar species in a cluster of that family

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C1	Lamiaceae(F)	3.19E-52	*Salvia adenophylla* Hedge & Hub.-Mor.; *Salvia aramiensis* Rech.f.; *Salvia atropatana* Bunge; *Salvia aucheri* Bentham; *Salvia blepharochlaena* Hedge & Hub.-Mor; *Salvia bracteata* Banks & Sol; *Salvia cadmica* Boiss.; *Salvia caespitosa* Montbret & Aucher ex Benth.; *Salvia candidissima* Vahl; *Salvia chionantha* Boiss.; *Salvia cilicica* Boiss.; *Salvia euphratica var. leiocalycina* (Rech.f.) Hedge; *Salvia frigida* Boiss.; *Salvia glutinosa* L.; *Salvia heldreichiana* Boiss.; *Salvia huberi* Hedge; *Salvia hypargeia* Fisch. & C.A.Mey.; *Salvia kronenburgii* Rech.f.; *Salvia macrochlamys* Boiss. & Kotschy; *Salvia microstegia*Boiss. & Balansa; *Salvia modesta* Boiss.; *Salvia multicaulis* Vahl; *Salvia nemorosa* L.; *Salvia pachystachys* Trautv; *Salvia pisidica* Boiss. & Heldr. ex Benth.; *Salvia potentillifolia* Boiss. & Heldr. ex Benth.; *Salvia recognita* Fisch. & C.A.Mey.; *Salvia rosifolia* Sm.; *Salvia russellii* Benth.; *Salvia sclarea* L.; *Salvia staminea* Montbret & Aucher ex Benth.; *Salvia suffruticosa* Montbret & Aucher ex Benth.; *Salvia syriaca* L.; *Salvia tomentosa* Mill.; *Salvia poculata* Nábelek; *Salvia verticillata* subsp. *amasiaca* (Freyn & Bornm.) Bornm.; *Salvia adenocaulon* P.H.Davis; *Salvia aethiopis*L.; *Salvia limbata* C.A.Mey.; *Salvia virgata* Jacq.; *Salvia palestina* Benth.	985;2537;2682;2758;3893;6549; 6616;6654;7361; 7463;8094;8222;10364;10494; 10582;11005;11008; 11463;11636;12405;12406;12409; 12412;12413; 12592;14529;29025;62367;64685; 85567;88302; 91354;91457;92231;93081; 121719; 122484;163263; 219891;289151;441005;519545; 519857;530816; 534398;535296;584507;591419; 1742210;5280435; 5280678;5281515;5320128; 5365034;5365586; 5367550;5373727;6429040; 6429301;6857681; 9855795;10104370;10704181; 12311096;12313020; 13894537;15560276; 42608158; Acetamide,N(2-acetyloxy)-2-[3,4-bis(acetyloxy)phenylethyl]-N-methyl; 5-(1-Isopropenyl-4,5-dimethylbicyclo [4,3,0]nonan-5yl)-3-methyl-2-pentanolacetate; 3-Isopropyl-6,7-dimethyltricyclo [4,4,0,0(2,8)]decane-9,10-diol;

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
				2(1H)Naphthalenone,3,5,6,7,8, 8ahexahydro-4,8a-dimethyl-6-(1- methylethenyl) ; Isochiapin B; Methyl (*Z*)-13-docosenoate
C2	Enterobacteriaceae (F)	2.38E-32	*E.Coli* (12); *E.Coli* (9); *E.Coli* (8); *K.pneumoniae* (4); *K.pneumoniae* (5); *K.pneumoniae* (6); *K.pneumoniae* (9); *E.Coli* (24); *K.pneumoniae* (13); *K.pneumoniae* (14); *K.pneumoniae* (15); *K.pneumoniae* (16); *K.pneumoniae* (17); *K.pneumoniae* (18); *K.pneumoniae* (7); *K.pneumoniae* (8); *E.Coli* (17); *E.Coli* (18); *E.Coli* (21); *E.Coli* (15); *E.Coli* (10); *E.Coli* (11); *E.Coli* (7); *E.Coli* (4); *E.Coli* (20); *E.Coli* (13); *E.Coli* (14); *E.Coli* (22); *K.pneumoniae* (3); *E.Coli* (19); *E.Coli* (16); *E.Coli* (1); *P.aeruginosa* (1); *K.pneumoniae* (1)	8174;8193;31260
C3	Burkholderiaceae(F)	5.87E-23	*Burkholderia glumae* LMG 2196; *Burkholderia glathei* LMG 14190; *Cellulomonas uda*; *Burkholderia lata* LMG 22485; *Burkholderia graminis* LMG 18924; *Serratia plymuthica* IC14; *Burkholderia anthina* LMG 20980; *Burkholderia caledonica* LMG 19076; *Burkholderia fungorum* LMG 16225; *Burkholderia caryophylli* LMG 2155; *Burkholderia caribensis* LMG 18531; *Escherichia coli*; *Burkholderia gladioli* LMG 2216; *Burkholderia pyrrocinia* LMG 21822; *Serratia marcescens* MG1; *Serratia entomophilia* A1MO2; *Burkholderia*	6054;7220;8103;8129;8163;11952; 13187; 13190; 23686;31260

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
			phytofirmans LMG 22487; *Burkholderia terricola* LMG 20594; *Serratia plymuthica* HRO-C48; *Pseudomonas fluorescens* WCS 417r; *Pseudomonas putida* ISOf; *Chromobacterium violaceum*; *Burkholderia lata* LMG 6993; *Chromobacterium violaceum* CV0; *Burkholderia sacchari* LMG 19450; *Serratia proteamaculans* B5a; *Burkholderia phenazinium* LMG 2247; *Burkholderia thailandensis* LMG 20219; *Burkholderia phenoliruptrix* LMG 22037; *Burkholderia xenovorans* LMG 21463; *Burkholderia cepacia* LMG 1222 358; *Burkholderia hospita* LMG 20598	
C4	Streptomycetaceae (F)	1.16E-40	*Streptomyces albidoflavus* AMI 246; *Streptomyces coelicolor* DSM 40233; *Streptomyces griseus* ATCC 23345; *Streptomyces hirsutus* ETH 1666; *Streptomyces rishiriensis* AMI 224; *Streptomyces albus*; *Streptomyces griseus* IFO 13849; *Streptomyces coelicolor* ATCC 21666; *Streptomyces thermoviolaceus* CBS 111.62; *Streptomyces coelicolor*; *Streptomyces murinus* NRRL 8171; *Streptomyces hygroscopicus* IFO 13255; *Streptomyces murinus* DSM 40091; *Streptomyces olivaceus* ETH 7437; *Streptomyces albidoflavus*; *Streptomyces hygroscopicus* ATCC 27438; *Streptomyces* spp. AMI 240; *Streptomyces antibioticus* CBS 659.68; *Streptomyces hirsutus* ATCC 19773; *Streptomyces antibioticus*; *Streptomyces antibioticus* ETH 22014; *Streptomyces aureofaciens* ETH 28832; *Streptomyces diastatochromogenes* ETH 18822; *Streptomyces aureofaciens* ETH 13387; *Streptomyces diastatochromogenes* IFO 13814; *Streptomyces olivaceus* ETH 6445; *Streptomyces albus* subsp. pathocidicus IFO 13812; *Streptomyces* spp. AMI 243; *Streptomyces albus* IFO 13014	180;263;6557;6560;8452;8723; 12232;12988; 19310;29746;31260

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C5	Leuconostocaceae (F)	3.87E-23	*Lactobacillus plantarum*; *Lactobacillus casei* NCIB 8010; *Lactobacillus plantarum* NCIB 6376; *Lactococcus lactis*; *Lactococcus lactis* DSM 20202; *Leuconostoc cremoris* DSM 20346; *Leuconostoc dextranicum* DSM 20484; *Leuconostoc mesenteroides* DSM 20343; *Leuconostoc oenos*; *Leuconostoc oenos* 19; *Leuconostoc oenos* 30; *Leuconostoc oenos* 36; *Leuconostoc oenos* 37D; *Leuconostoc oenos* 7B; *Leuconostoc oenos* B66; *Leuconostoc oenos* DSM 20252; *Leuconostoc oenos* DSM 20255; *Leuconostoc oenos* DSM 20257; *Leuconostoc oenos* Lc5x; *Leuconostoc paramesenteroides* DSM 20288; *Pediococcus damnosus* DSM 20331	176;179;240;244;264;379;2969; 3893; 6054; 6560;7344;7991;8094;8158;8892; 10448; 11005;11732;12053;31260
C6	Prevotellaceae (F)	1.65E-18	*Prevotella heparinolyticus* ATCC 35895; *Bacteroides fragilis* ATCC 25285; *Porphyromonas endodontalis* HG 181 (H 11a-e); *Porphyromonas endodontalis* HG 182 (BN 11a-f); *Porphyromonas endodontalis* HG 370 (ATCC 35406); *Porphyromonas endodontalis* HG 412; *Prevotella oralis* ES14B-3A; *Prevotella oralis* ES15-2; *Prevotella oris* RPG; *Prevotella veroralis* ATCC 33779; *Prevotella buccae* ATCC 33574; *Prevotella buccae* ES12-B; *Prevotella buccae* ES17-1; *Prevotella buccae* ES9-1; *Prevotella disiens* DSM 20516; *Prevotella oralis* ES4-B; *Prevotella oris* ATCC 33573; *Prevotella oris* ES14B-3A; *Prevotella oris* ES9-3; *Bacteroides fragilis*	985;11005;21205;21672;33002; 151014; 161495; 164860;182089;301590;520298
C7	Cannabaceae (F)	1.05E-15	*Humulus lupulus* L. ' US Northern Brewer hops ' ; *Humulus lupulus* L. ' German Hallertau hops ' ; *Humulus lupulus* L. ' Kent Golding hops ' ; *Ficus perforata* L.; *Humulus lupulus* L. ' Citra hops ' ; *Humulus lupulus* L. ' Cascade hops ' ; *Pilea nummularifolia* (Sw.) Wedd.; *Humulus lupulus* L. ' Sirachi Ace hops ' ; *Trema micrantha* (L.) Blume; *Humulus lupulus* L. ' Czech Saaz hops ' ; *Humulus lupulus* L. ' Williamette hops ' ; *Humulus lupulus* L. ' Chinook hops ' ; *Humulus lupulus* L. ' Columbus hops ' ; *Lantana involucrata* L.;	6654;14896;22311; 31253; 5281515; 5281520

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
			Rhaphiolepis umbellata Makino ; *Salvia coccinea* Buc'hoz ex Etl.; *Salvia miniata* Fernald	
C8	Hypocreaceae (F)	1.92E-27	*T. asperellum* GJS 02-65; *T. asperellum* CBS 433.97; *T. aggressivum* DAOM222156; *T. aggressivum* IMI 393970; *T. stromaticum* GJS 00-127; *T. virens* DAOM167651; *H. koningii* CBS 989.97; *T. viride* GJS 04-379; *T. pseudokoningii* CBS 480.91; *T. atroviride* CBS 351.93; *T. viride* BBA 70239; *T. inhamantum* CBS 273.78; *T. pseudokoningii* CBS 130756; *T. longibrachiatum* CBS 118642; *T. longibrachiatum* TR97; *T. harzianum* CBS 227.95; *T. brevicompactum* CBS 109720	176;179;180;243;261;454;650;702; 2537;3776; 6276;6560;6561;6569;6590;7284; 7501;7720; 8051;8063;8158;8175;8723;8857; 8892;11552; 13187;14257;22311;31260;31289; 86608;246728; 5281516;5281517;5281522; 5316209;Xylene(M-P-and O-)
C9	Bacteroidaceae (F)	9.46E-05	*Porphyromonas gingivalis* FDC381; *Porphyromonas gingivalis* W83; *Prevotella loescheii*; *Prevotella loescheii* ATCC 15930; *Clostridium sporogenes*; *Bacteroides ovatus*; *Capnocytophaga ochracea* ATCC 33596; *Clostridium bifermentans*; *Bacteroides distasonis*; *Bacteroides thetaiotamicron*; *Bacteroides vulgatus*; *Prevotella intermedia* ATCC 25261; *Porphyromonas gingivalis*; *Fusobacterium nucleatum*; *Veillonella* spp.	176; 1032; 6590; 10430
C10	Rivulariaceae(F)	3.82E-10	*Calothrix parietina*; *Plectonema* sp.; *Plectonema*; *Calothrix*; *Plectonema notatum*; *Phormidium* sp.; *Tolypothrix*; *Tolypothrix distorta*; *Calothrix* sp.; *C. parietina*; *Rivularia* sp.;biofilms A (*Rivularia* sp./*C. parietina* community); *Cyanobacteria*	454;8175;9862; 9895;12398;22311;30398;31289; 88332; 292723;519382;638014;5352481

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C11	Trichocomaceae (F)	1.47E-13	*Aspergillus flavus* NRRL 18543; *Aspergillus flavus*; *Aspergillus flavus* NRRL 25347; *Aspergillus niger*; *Aspergillus niger* NRRL 326; *Aspergillus parasiticus* NRRL 5862; *Penicillium glabrum* NRRL 766; *Rhizopus stolonifer*; *Rhizopus stolonifer* NRRL 54667	957;1031;6184;6276;8051;8093; 8103;8129; 8452;10886;10976;11428;11527; 11583;12020; 12297;12756;13187;13561;18827; 19602;20083; 20534;22386;61177;138644; 246728;5281167; 5283316;5283324;5318599; 5352837;5364631; 5364919;11116296; 5-Octen-2-ol; Chalcogran(*Z*)
C12	Bacillaceae (F)	3.65E-06	*Streptomyces lateritius*; *Bacillus simplex*; *Bacillus subtilis*; *Bacillus weihenstephanensis*; *Microbacterium oxydans*; *Stenotrophomonas maltophilia*; *Serratia marcescens*	180;240;996;998;1032;2682;6054; 7410;8079;8141;8163;8175;8182; 11006;12232;12389;13187;17100; 20083;26808
C13	Helicobacteraceae(F)	2.09E-06	*Bacteroides ureolyticus* CCUG 7319 (ATCC 33387); *Bacteroides gracilis* CCUG 13143 (ATCC 33236); *Campylobacter fetus* subsp. *venerealis* CCUG 538 (ATCC 19438); *Wolinella curva* CCUG 13146 (ATCC 35224); *Wolinella recta* FDC 371 (ATCC 33238); *Wolinella succinogenes* CCUG 12550 (ATCC 29543)	985;3893;5281;11005;16064;301590
C14	Paxillaceae(F)	0.000201	*Laccaria bicolor* (Maire) P.D.Orton ; *Armillaria mellea* (Vahl) P.Kumm.; *Paxillus involutus* MAJ; *Paxillus involutus* NAU; *Stropharia rugosoannulata* Farlow ex Murrill; *Pholiota squarrosa* (Oeder) Kumm.	179;356;6549;7394;8723;15095; 16319;18827; 19602;31260;86895;102861; 246728;441005; 442393;595137;3033866;5281517; 5281520; 5284507;5373727;6429077; 6918391; 10104370; 12306052

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C15	Xanthomonadaceae(F)	0.000201	*Arthrobacter nitroguajacolius*; *Lysobacter gummosus*; *Sporosarcina ginsengisoli*; *Alcaligenes*; *Alcaligenes faecalis*; *Stenotrophomonas maltophilia*	178;240;998;6329;7222;7976;8007; 8182;12232; 12401;13381;31252
C16	Ophiostomataceae(F)	1.63E-08	*E. polonica*; *G. europhioides*; *O. minus*; *O. piceae*; *G. penicillata*; *O. bicolor*	6054;8103;31260;31276
C17	Desulfovibrionaceae(F)	0.000201	*Desulfovibrio gigas*; *Clostridium collagenovorans*; *Desulfovibrio vulgaris*; *Methanosarcina barkeri*; *Methanobacterium formicicum*; *Scopulariopsis brevicaulis*	11648;11656;23496;68977;68978
C18	Enterobacteriaceae (F)	4.49E-05	*Serratia plymuthica* 4Rx13; *Serratia plymuthica* 3Re-4-18; *Serratia plymuthica* AS9; *Serratia plymuthica* HRO-C48; *Serratia plymuthica* V4	6054;6569;8163;8918;9261;11583; 12232;13187;19310;46187193; Diethyl ester; Hexadecen-1-ol; N-Heptadecyl Ester
C19	Rhodobacteraceae(F)	4.26E-08	*Sulfitobacter pontiacus*; *Sulfitobacter pontiacus* BIO-007; *Sulfitobacter dubius* BIO-205; *Sulfitobacter* sp.; *Loktanella hongkongensis* strain Bio-204	14296;25915;25916;61700;519564; 521013; 11658321; 2-(2-Methylbutyl)-3,6-dimethylpyrazine; 2-Isobutyl-3,6-dimethylpyrazine
C20	Rhodobacteraceae(F)	1.48E-06	*Dinoroseobacter shibae* strain DFL-27; *Dinoroseobacter shibae*; *Loktanella* sp.; *Loktanella* sp. Bio-204	6054;7704;7710;7714;7742;8186; 8794; 9862;12756;12813;14296;16821; 18064; 19310;25916;93236;241520; 521013; 1549778;5352428;18466382; 87786981
C21	Pseudomonadaceae (F)	1.63E-08	*Pseudomonas chlororaphis*; *Pseudomonas aurantiaca*; *Pseudomonas corrugate*; *Pseudomonas fluorescens*	240;7222;7720;7966;7976;8141; 8163; 8174;8182;9261;11006; 11622; 12389; 12391;12401;13190;14257;15076; 15600; 19147;19310;31289;

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
				Phenylenediamine
C22	Staphylococcaceae(F)	1.63E-08	*S.aureus* (17); *S.aureus* (11); *S.aureus* (20); *S.aureus* (8)	24020;31260;31276
C23	Saccharomycetaceae(F)	0.095283	*Saccharomyces cerevisiae* Y1001; *Saccharomyces cerevisiae*; *Serratia* spp. B2675; *Serratia* spp. B675	176;180;702;1017;3776;6560; 6569;6590; 7237;7720;7895;7904;8857; 10430;11552; 12232;14257;22311;31252
C24	Rhodobacteraceae(F)	1.48E-06	*Oceanibulbus indolifex* HEL-45; *Roseobacter gallaeciensis* strain PIC-68; *Sulfitobacter* sp. PIC-70; *Loktanella hongkongensis* strain Bio-204	25915;25916;61700
C25	Xylariaceae(F)	1.09E-09	*Muscodor fengyangensis* (ZJLQ070); *Muscodor fengyangensis* (ZJLQ151); *Muscodor fengyangensis* (ZJLQ374); *Muscodor fengyangensis* (ZJLQ024)	289151;5317844;9855795; 10856614
C26	Amaryllidaceae (F)	1.04E-07	*A. fistulosum* L.; *A. chinense* G.Don; *A. tuberosum*Rottler ex Spreng.	9258;12377;12401;16590;19310; 22383;34296;68135;139446; 150636;587784;5283345;5320722; 5362897; Di-S-b, c-unsaturated thiosulfinates; Fully saturated thiosulfinates; Mixed a, b- and c-unsaturated thiosulfinates; Mono-a, b-unsaturated thiosulfinates
C27	Burkholderiaceae(F)	0.00024	*Burkholderia ambifaria* LMG 19182; *Burkholderia ambifaria* LMG 17828; *Burkholderia ambifaria* LMG 19467	1068;7410;7909;8163;11363;11622;13187;13576;14296;14505; 18064; 19310;26808;31246;31252;61700; 61922;93236;247573;1549778;

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
				1711945;53427438;3-Methylbutan-4-olide; Dodecen-2-one; Tetradec-8-en-2-one
C28	Myxococcales(O)	1.04E-07	*Chondromyces crocatus* Cm c2; *Chondromyces crocatus*; *Chondromyces crocatus* Cm c5	244;4133;6054;7222;7409;7476; 8499;9016; 11952;17898;31252;31289;32594; 33166; 34590;61151;61700;68255;519857; 520098; 586728;6428338;6428488; 10609095;11491990; 11637154;11679913;12722500; 15014537; 21037377;67080977;129866837; (1R,6S,10S)-6,10-Dimethylbicyclo[4.4.0]decan-3-one; 2-(1-hydroxy-1-methylethyl)-3-methoxypyrazine; 2-(1-hydroxy-1-methylpropyl)-3-methoxypyrazine; 2-(1-methylethenyl)-6-(1-methylethyl)pyrazine; 2-Butan-2-yl-3-methoxy-5-isobutylpyrazine; 2-Isobutan-5-propan-2-yl pyrazine; 2-isopropyl-3-methoxy-5-butan-2-yl-pyrazine; 2-Isopropyl-3-methoxy-5-isobutylpyrazine;2-Isopropyl-5-isopropenylpyrazine; 2-Methoxy-3-(1-hydroxy-2-methylpropyl)pyrazine; 2-methylpropyl)pyrazine; 2-

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
				Methyl-3-methoxy-5-isobutylpyrazine; 3-Benzyl-2-methoxypyrazine; 3-Methoxy-2,5-diisopropylpyrazine; 3-Methoxy-2,6-dimethylpyrazine; Dimethyl-(1-methylethyl)pyrazine
C29	Lactobacillaceae (F)	0.001113	*Lactobacillus hilgardii*; *Lactobacillus brevis*; *Oenococcus oeni*	878;10448;12232;18050;18635
C30	Liliaceae(F)	1.04E-06	*Lilium sargentiae*E.H.Wilson;*Lilium regale*E.H.Wilson;*Lilium sulphureum*Baker ex Hook.f.	240;454;998;2345;2758;4133;6054; 6549;6616;6654;7150;7193;7460; 7461;7462;7463;7654;7794;8785; 9862;11230;11463;14896;17100; 17868;18818;22311;31253;31289; 62385;176983;5281516;5281553; 5320250;5368451;5368821; 5371125;12315151

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C31	Nannocystaceae(F)	1.04E-07	*Nannocystis exedens* Na eB37; *Nannocystis exedens*; *Nannocystis exedens* subsp. cinnabarina Na c29	244;2682;6054;7222;7342;7361; 7742;7758; 7921;7945;8193;8209;8794;10128; 11902; 11952;12756;16913;22311;24020; 29746; 31249;64685;80148;155902; 247573;5373727; 6428338;10609095;11491990; 11571943; 25244198;129866837; (*E*)-beta methyl geraniol; 2,5-Dimethyl-3-(1-Methylethyl)pyrazine; 2-Isopropyl-5-iso propenyl pyrazine; 5-(1-methylethyl)-2-(2-methylpropyl) pyrazine
C32	Poaceae(F)	1.04E-07	*Oryza sativa* L. (BA-370);*Oryza sativa* L. (AM-157);*Oryza sativa* L. (IR-64)	240;243;332;454;460;957;996;998; 1183;4133;6184;6276;6448;6549; 7438;7463;7720;7792;7797;7799; 7800;7809;8051;8063;8103;8130; 8141;8163;8175;8182;8635;9231; 9862;9895;10703;12389;12391; 12398;12401;13187;13572;14079; 14260;17000;18827;19602;25209; 26049;31236;31265;31289;61124;

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
				62374;75027;85779;91354;95495; 97960;102562;288227;289151; 439250;522834;545551;643731; 5281167;5281168;5281515; 5282108;5283324;5283329; 5283335;5283339;5283349; 5362616;5362716;5363229; 5364450;5364594;5364941; 6918391; (*E*)-5-Ethyl-6-methyl-3-hepten-2-one; Pinene
C33	Enterobacteriaceae (F)	0.052301	*Salmonella enterica*; *Candida tropicalis*; *Shigella flexneri*	176;177;180;263;702;878;887; 6569;8723
C34	Cucurbitaceae(F)	1.04E-07	*Solena amplexicaulis* (Lam.) Gandhi;*Momordica cochinchinensis* (Lour.) Spreng.;*Solena amplexicaulis* (Lam.) Gandhi	244;957;6654;8051;8103;8129; 8175;8184;8193;8207;8221;11527; 12178;12297;12397;18818;18827; 22310;31289;62388;246728; 637566;643820;5281516;5281553; 5282108;5318042;5365872

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C35	Cystobacteraceae(F)	4.17E-07	*Stigmatella aurantiaca* DW4/3-1; *Stigmatella aurantiaca*; *Stigmatella aurantiaca* Sg a15	240;244;2682;6054;6835;7150; 7302;7410;7921;8163;8209;9862; 10285;10430;11529;11558;12187; 14505;15229;17100;19310;27209; 29746;31272;78167;80108;80148; 91457;94094;97746;98643;103851; 118823;139108;254132;278218; 445070;528743;1549778;5284507; 5323652;5373727;10609095; 12302138;25244198;57339298; 87882425; 2,12-Dimethyltridecan-4-one; 2-Methyltetradecan-4-one; 2-Methyltridec-2-en-4-one; 2-Methyltridecan-4-one; 3-Methyltridecan-4-one; 9-Methyl-1-phenyldecan-1-one
C36	Araceae(F)	1.04E-07	*Homalomena debilicrista* Y.C.Hoe; *Homalomena velutipedunculata* S.Y.Wong, Y.C.Hoe & P.C.Boyce.; *Homalomena gastrofructa* S.Y.Wong, Y.C.Hoe & P.C.Boyce.	6654;7043;7758;8051;10976; 14896;17868;18818;22311;31253; 5320250;6427110;12589924
C37	Burkholderiaceae(F)	0.00024	*Burkholderia kururiensis* LMG 19447; *Burkholderia phenazinium* LMG 2247; *Burkholderia xenovorans* LMG 21463	244;6054;8103;8129;8163;12741; 19310;31260

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C38	Lamiaceae(F)	0.001382	*Salvia trichoclada* Benth.;*Salvia aethiopis* L.;*Salvia limbata* C.A.Mey.	985;2537;2682;2758;3893;6448;6616;6654;8094;10364;10494;10819;11005;11008;11636;12405;12406;12408;12409;12412;12413;12534;12592;14529;29025;64685;85567;88302;91354;91457;92231;117452;121719;122484;163263;188457;219891;289151;441005;442393;519545;519857;530432;530816;534398;535296;575363;584507;591419;1742210;5280435;5280794;5281515;5320128;5365034;5365586;5367550;6434253;6857681;10104370;10704181;12311096;12313020;13894537;14165786;15560276;42608158; Acetamide,N(2-acetyloxy)-2-[3,4-bis (acetyloxy) phenyl] ethyl]-Nmethyl; 5-(1-isopropenyl-4,5-dimethylbicyclo [4,3,0]nonan-5-yl)-3-methyl-2-pentanolacetate; 3-Iso-propyl-6,7-dimethyltricyclo [4,4,0,0(2,8)] decane-9,10-diol; 2(1H)Naphthalenone,3,5,6,7,8,8ahexahydro-4,8a-dimethyl-6-(1-methylethenyl); Isochiapin B; Methyl (Z)-13-docosenoate; Stearic acid, allyl ether

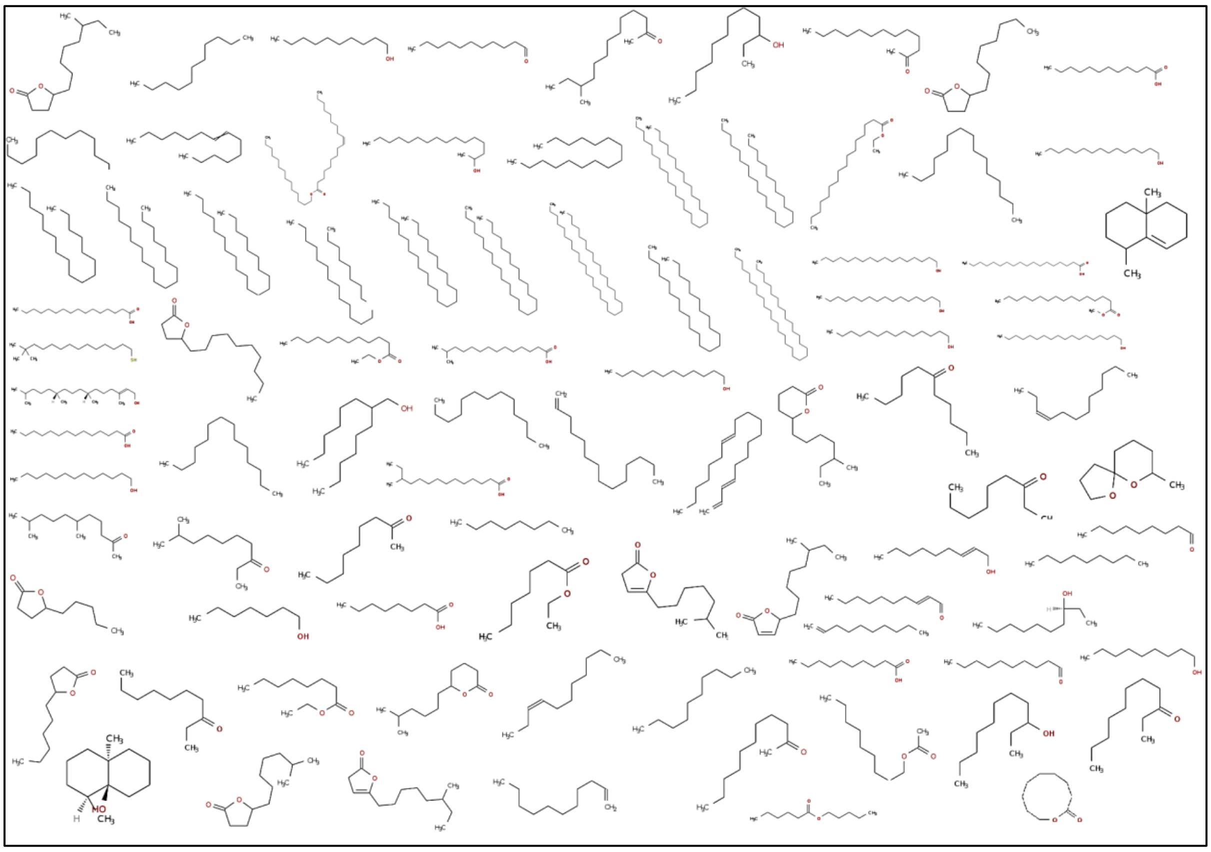
CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C39	Tuberaceae(F)	1.04E-07	*Tuber rufum* Pico, 1788; *Tuber brumale* Vitt; *Tuber simonea*	177;180;263;280;702;1068;3776; 6560;6568;6569;6584;7749;7762; 7770;7803;8723;8857;12180; 13357;18829;24020
C40	Veillonellaceae(F)	0.040969	*Bacteroides fragilis*; *Veillonella* spp.	176;1110;6590;10430
C41	Meruliaceae(F)	1.34E-05	*Bjerkandera adusta* CBS 595.78; *Bjerkandera adusta*	125;126;240;243;244;7118;7410; 7738;8419;31244
C42	Enterobacteriaceae (F)	0.019159	*E.Coli* (15); *K.pneumoniae* (11)	264;8209;31260
C43	Enterobacteriaceae (F)	0.019159	*Enterobacter* spp.; *Citrobacter* sp.	6054;10448;12232;78925
C44	Gammaproteobacteria(C)	0.005168	Gamma proteobacteria; Alpha proteobacteria	5281;8158; Carboxylic Acid Methyl Esters
C45	Streptococcaceae(F)	0.040969	*Lactococcus* sp.; *Lactobacillus* sp.	402;878;1068;12232;19310
C46	Liliaceae(F)	0.000134	*Lilium primulinum var. ochraceum* (Franch.) Stearn;*Lilium bakerianum var. delavayi* (Franch.) E.H.Wilson	6549;6654;7150;7463;31253; 31289;5281553;5320250;5368451; 5371125
C47	Myxococcaceae(F)	0.005168	*Myxococcus xanthus*;*Myxobacterium* spp.	9862;10285;11952;13576;29746; 61235;75189; 79828;98970;246728;445713; 1549778;5373727; 10609095;10654501;21278276; 9-Methyl-3-decanol; Octalin Hydrocarbon
C48	Sordariaceae(F)	1.34E-05	*Neurospora* sp.; *Neurospora sitophila* ATCC 46892	702;8857;18827;31260;31265
C49	Rhodobacteraceae(F)	0.001406	*Octadecabacter* sp. ARK10255b; *Octadecabacter* sp.	240;14505;31252;61922;7000073
C50	Asparagaceae(F)	1.34E-05	*Ornithogalum sigmoideum* Freyn & Sint.;*Ornithogalum orthophyllum* Ten.	240;454;957;998;6184;8029;8042; 8103;8129; 8130;8141;8175;8181;8914;11006; 12366;12388; 12389;12391;12401;12403;14257; 19602;31289;

CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
				79523;95568;5283324;5283345 ; Hexamethylpyranoindane
C51	Paenibacillaceae(F)	1.34E-05	*Paenibacillus polymyxa* E681; *Paenibacillus polymyxa*	179;180;262;263;650;702;760;878; 6328; 6557;6560;6569;6915;8182;8723; 8857; 10430;11552;12232;12388;13190; 14257; 14489;19310;19602;31260;31272; 521790;637564
C52	Pseudomonadaceae(F)	1.34E-05	*Pseudomonas putida* ISOf; *Pseudomonas putida*	244;798;3893;7220;7362;7731; 8129; 8163;11005;11952;12232;13187; 13190; 14257;19310;21210;31260
C53	Enterobacteriaceae (F)	0.030807	*S.aureus* (14); *K.pneumoniae* (12)	264;11552;31260
C54	Enterobacteriaceae (F)	0.030807	*S.aureus* (4); *E.Coli* (5)	176;177;264;650;6569;7762;7895; 8051; 8857;31260;31265
C55	Cystobacteraceae(F)	0.076114	*Stigmatella* sp.; *Dinoroseobacter* sp.	7302;7704;7710;7714;7742;7921; 12756; 12813;16821;97746
C56	Streptomycetaceae (F)	0.006641	*Streptomyces* sp. GWS-BW-H5.; *Streptomyces caviscabies*	10285;11173;11952;12988;13572; 28965 ;155902;441005;519857; 521496; 6427091; 6427110;6429077;10609095; 12306047; 15560276;21778193;21778194; 21778195; 21778196;21778197;21778198; 21778199; 21778200;25203009;70678558; 88931680; 91723677;91749842

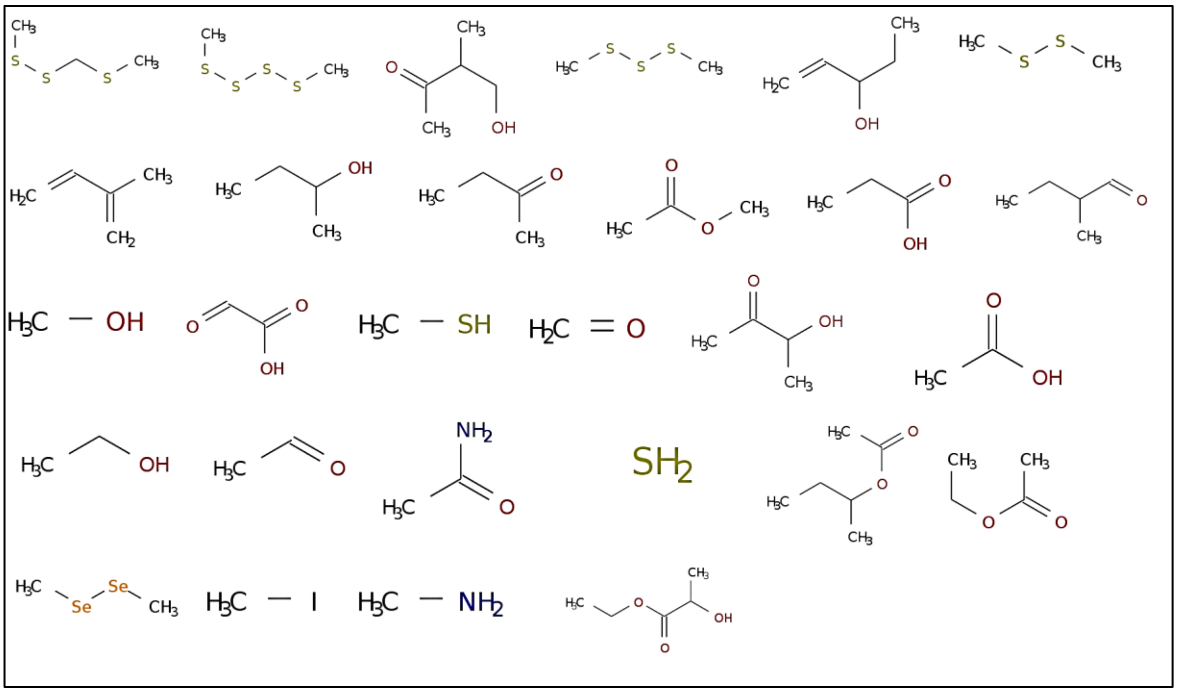
CID	Dominant Taxonomic Rank	p-value	Species Name	PubchemID(VOC)
C57	Ceratocystidaceae(F)	1.34E-05	*Thielaviopsis basicola*; *Ceratocystis* sp.	180;702;712;6561;7284;7362; 8051; 8857;5281168

Supplementary Figure 1

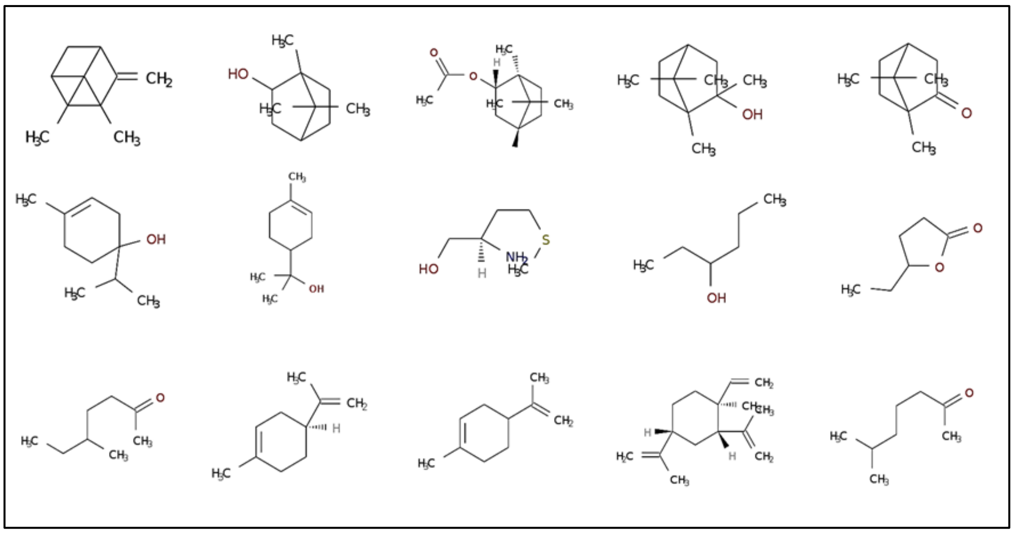
Structurally similar VOC group (SSVG) clusters (First four clusters)



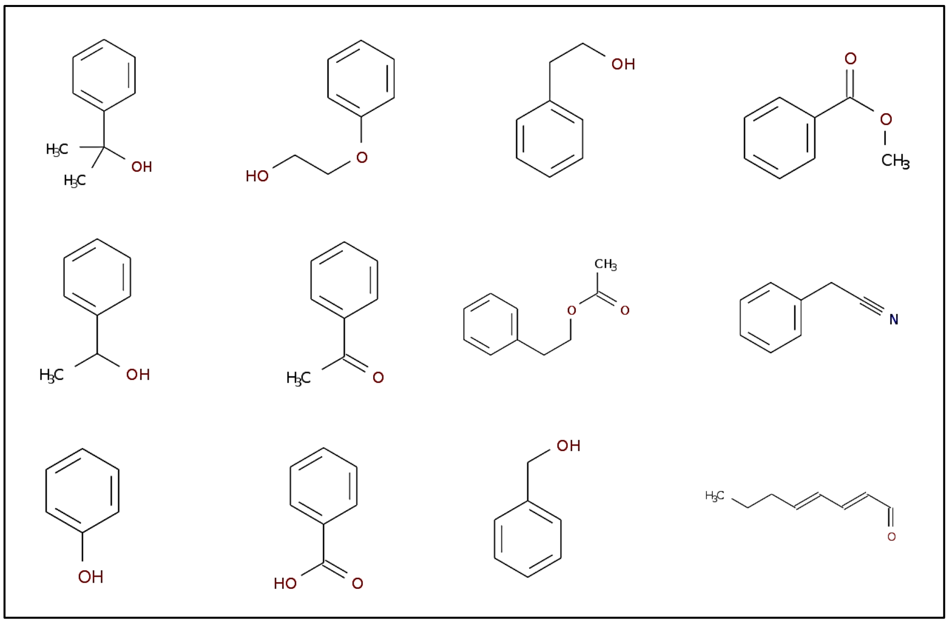
Cluster 1 (long chain hydrocarbon)



Cluster 2（Alcohol）



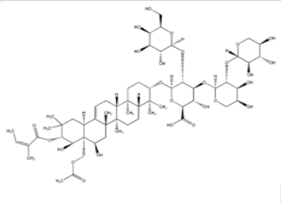
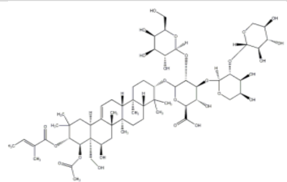
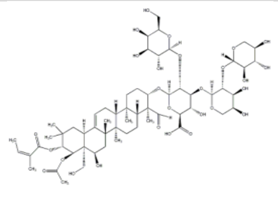
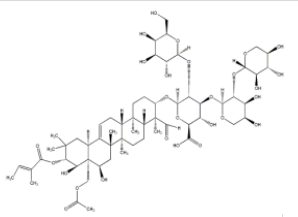
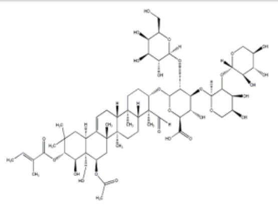
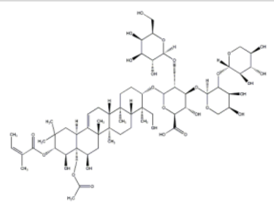
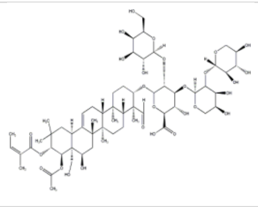
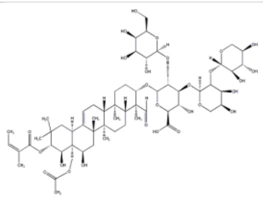
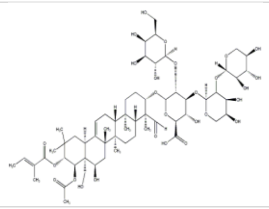
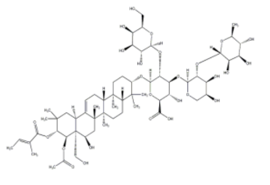
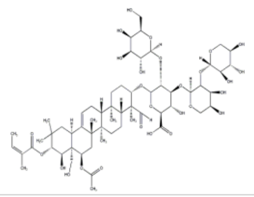
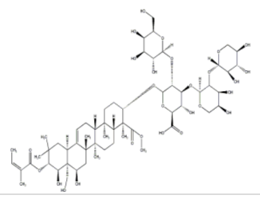
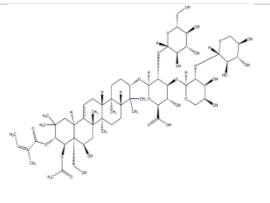
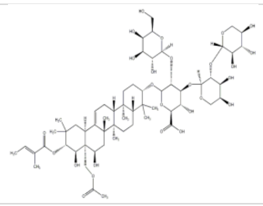
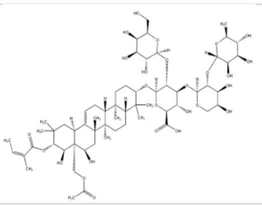
Cluster 3（Ring compound）



Cluster 4 (Phenolic compound)

Supplementary figure 2.

The list of top 6 clusters generated by DPClusOST on KNApSAcK metabolites.

		
C00029754 Assamsaponin E $C_{59}H_{92}O_{26}$	C00029931 Chakasaponin I $C_{59}H_{92}O_{26}$	C00029753 Assamsaponin D $C_{59}H_{92}O_{27}$
		
C00029752 Assamsaponin C $C_{59}H_{90}O_{27}$	C00032335 Theasaponin E8 (+)-Theasaponin E8 $C_{59}H_{90}O_{27}$	C00032322 Theasaponin A2 (+)-Theasaponin A2 $C_{59}H_{92}O_{27}$
		
C00032328 Theasaponin E1 $C_{59}H_{90}O_{27}$	C00032329 Theasaponin E2 (+)-Theasaponin E2 $C_{59}H_{90}O_{27}$	C00032334 Theasaponin E7 (+)-Theasaponin E7 $C_{59}H_{90}O_{27}$
		
C00030275 Floratheasaponin G $C_{60}H_{94}O_{26}$	C00032331 Theasaponin E4 (+)-Theasaponin E4 $C_{59}H_{90}O_{27}$	C00032337 Theasaponin F1 (+)-Theasaponin F1 $C_{58}H_{90}O_{27}$
		
C00030269 Floratheasaponin A $C_{59}H_{92}O_{26}$	C00029935 Chakasaponin VI $C_{59}H_{92}O_{26}$	C00030277 FloratheasaponinI (+)FloratheasaponinI $C_{60}H_{94}O_{26}$

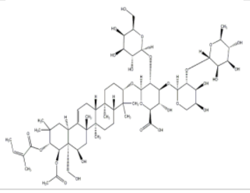
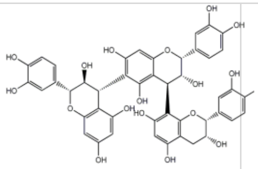
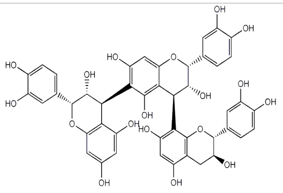
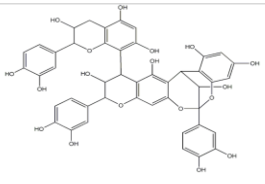
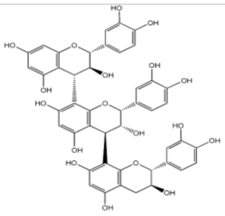
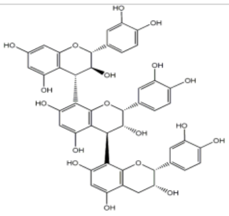
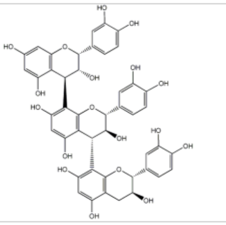
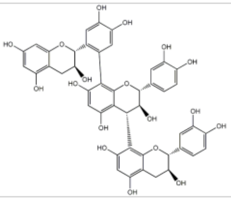
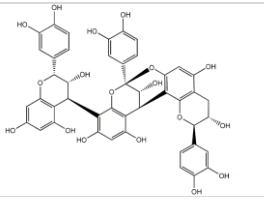
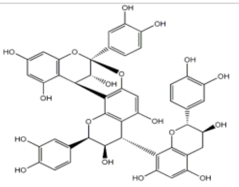
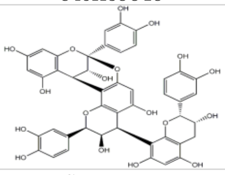
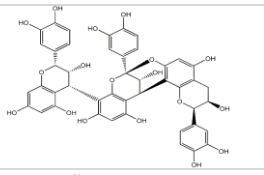
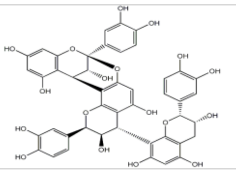
		
C00030272 Floratheasaponin D(-)-Floratheasaponin D $C_{60}H_{94}O_{26}$		

Figure S1(a): Cluster 1, 16 Elements, Triterpenoid saponins

		
C00009085 Catechin-(4alpha->6)-epicatechin-(4beta->8)-epicatechin $C_{45}H_{38}O_{18}$	C00009086 Epicatechin-(4beta->6)-epicatechin-(4beta->8)-catechin $C_{45}H_{38}O_{18}$	C00013282 Pavetannin B 5 $C_{45}H_{36}O_{18}$
		
C00009093 Catechin-(4alpha->8)-epicatechin-(4beta->8)-catechin $C_{45}H_{38}O_{18}$	C00009094 Catechin-(4alpha->8)-epicatechin-(4beta->8)-epicatechin $C_{45}H_{38}O_{18}$	C00009095 Procyanidin C4 $C_{45}H_{38}O_{18}$
		
C00008914 Catechin-(6'->8)-catechin-(4alpha->8)-catechin $C_{45}H_{38}O_{18}$	C00009299 Epicatechin-(4beta->8)-epicatechin-(2beta->7,4beta->8)-catechin $C_{45}H_{36}O_{18}$	C00009289 Cinnamtannin D1 $C_{45}H_{36}O_{18}$
		
C00009290 Epicatechin-(2beta->7,4beta->8)-epicatechin-(4alpha->8)-epicatechin $C_{45}H_{36}O_{18}$	C00009298 Epicatechin-(4alpha->8)-epicatechin-(2beta->7,4beta->8)-epicatechin $C_{45}H_{36}O_{18}$	C00009291 Cinnamtannin B1 $C_{45}H_{36}O_{18}$

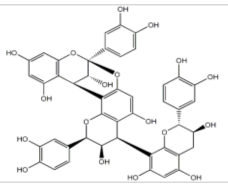
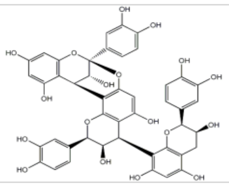
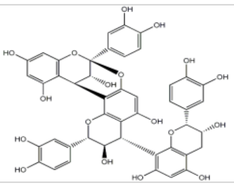
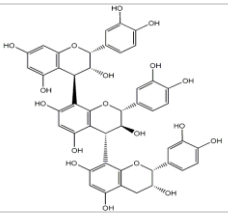
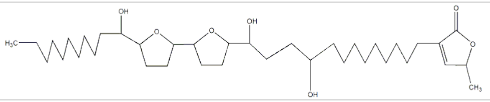
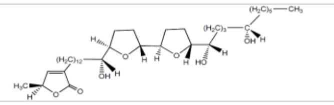
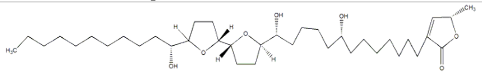
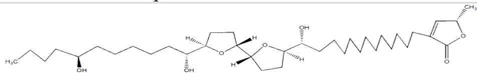
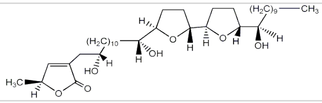
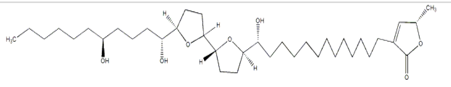
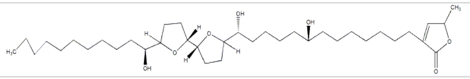
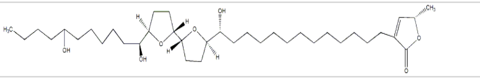
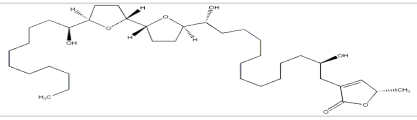
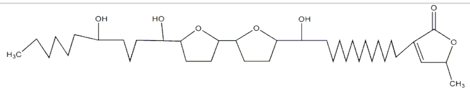
		
C00009288 Pavetannin B6 $C_{45}H_{36}O_{18}$	C00009292 Pavetannin B 1 $C_{45}H_{36}O_{18}$	C00009293 Aesculitannin B $C_{45}H_{36}O_{18}$
		
C00009096 Epicatechin-(4beta->8)-catechin-(4alpha->8)-epicatechin $C_{45}H_{38}O_{18}$		

Figure S1(b): Cluster 2, 16 Elements, Proanthocyanidins or Condensed tannins

	
C00044321 Squamocin F $C_{37}H_{66}O_7$	C00001323 Squamocin $C_{37}H_{66}O_7$
	
C00044098 Asimin $C_{37}H_{66}O_7$	C00044102 Asiminocin (+)-Asiminocin $C_{37}H_{66}O_7$
	
C00001301 Asimicin $C_{37}H_{66}O_7$	C00044103 Asitribin $C_{37}H_{66}O_7$
	
C00044117 Bullatin $C_{37}H_{66}O_7$	C00044117 Bullatin $C_{37}H_{66}O_7$
	
C00038662 Bullatacin (+)-Bullatacin $C_{37}H_{66}O_7$	C00044204 Isorollinicin $C_{37}H_{66}O_7$

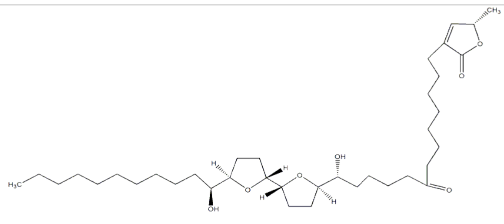
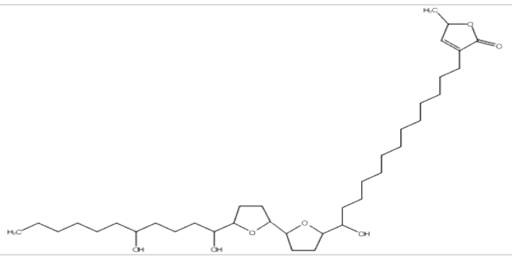
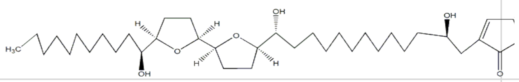
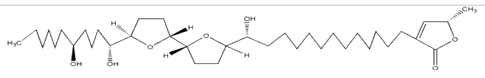
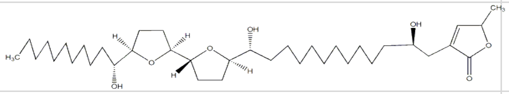
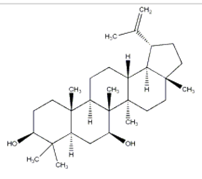
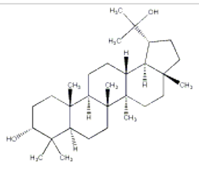
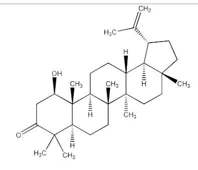
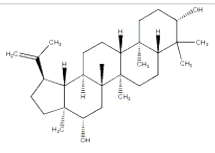
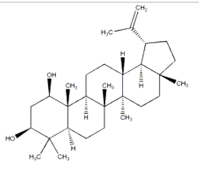
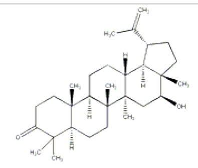
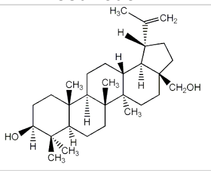
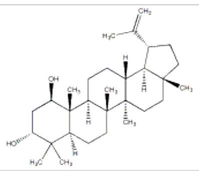
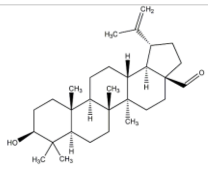
	
C00044190 Guanacone　$C_{37}H_{64}O_7$	C00049779 Carolin A $C_{37}H_{66}O_7$
	
C00044300 Rolliniastatin 1 $C_{37}H_{66}O_7$	C00044320 Squamocin D $C_{37}H_{66}O_7$
	
C00044359 Trilobacin　$C_{37}H_{66}O_7$	

Figure S1(c): Cluster 3, 15 Elements,　Acetogenin

		
C00033122 Loranthol　$C_{30}H_{50}O_2$	C00041268 3alpha,20-Lupandiol (-)-3alpha,20-Lupandiol $C_{30}H_{52}O_2$	C00023789 Glochidonol　$C_{30}H_{48}O_2$
		
C00034033 Lup-20(29)-ene-3beta,16beta-diol (+)-Lup-20(29)-ene-3beta,16beta-diol $C_{30}H_{50}O_2$	C00037440 Lup-20(29)-ene-1beta,3beta-diol $C_{30}H_{50}O_2$	C00034197 Resinone $C_{30}H_{48}O_2$
		

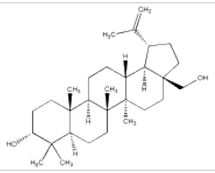
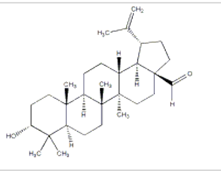
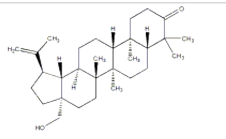
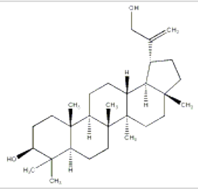
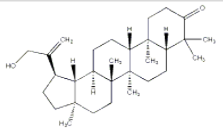
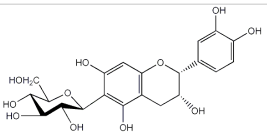
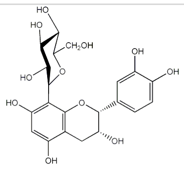
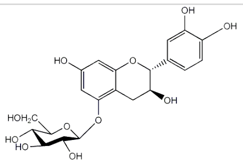
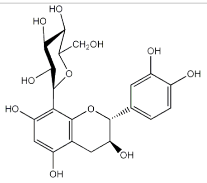
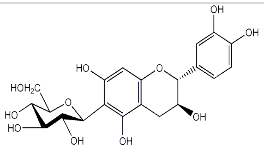
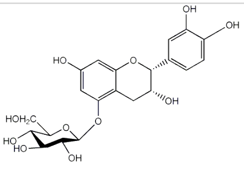
C00003740 Betulin C30H50O2	C00042563 Glochidiol C30H50O2	C00021224 Betulinaldehyde Betunal (+)-Betunal C30H48O2
		
C00042123 3-epi-Betulin (+)-3-epi-Betulin C30H50O2	C00042124 3-epi-Betulinic aldehyde C30H48O2	C00042283 Betulone (+)-Betulone C30H48O2
		
C00034871 Lup-20(29)-ene-3beta,30-diol (-)-Lup-20(29)-ene-3beta,30-diol C30H50O2	C00041266 30-Hydroxylup-20(29)-en-3-one (+)-30-Hydroxylup-20(29)-en-3-one C30H48O2	

Figure S1(d): Cluster 4, 14 Elements, Triterpene

		
C00006356 6-C-Glucopyranosylepicatechi n C21H24O11	C00006357 8-C-Glucopyranosylepicatechin C21H24O11	C00008846 Catechin-5-O-beta-D-glucopyranoside (+)-Catechin-5-O-beta-D-glucopyranoside C21H24O11
		
C00006359 8-C-Glucopyranosylcatechin C21H24O11	C00006358 6-C-Glucopyranosylcatechin C21H24O11	C00008847 Epicatechin-5-O-beta-D-glucopyranos ide (-)-Epicatechin-5-O-beta-D-glucopyranosid e C21H24O11

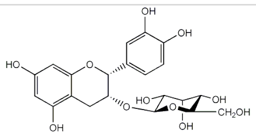
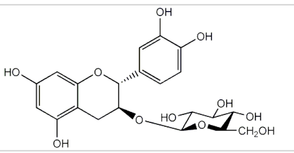
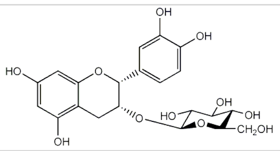
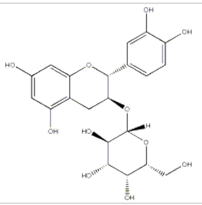
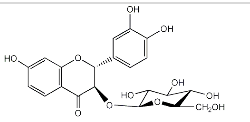
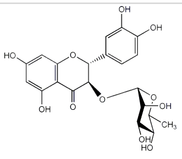
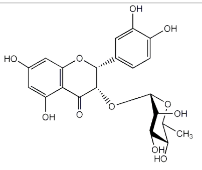
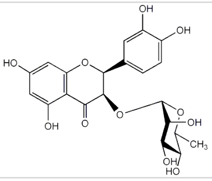
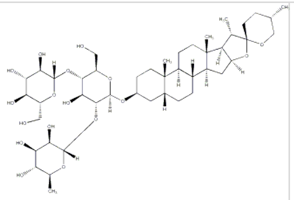
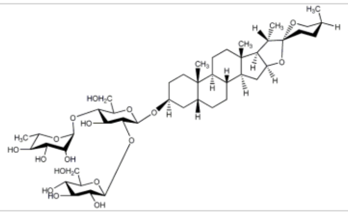
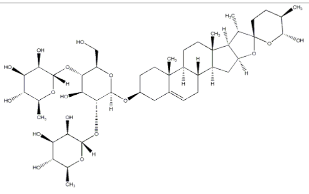
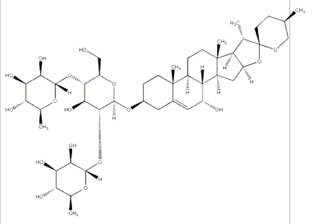
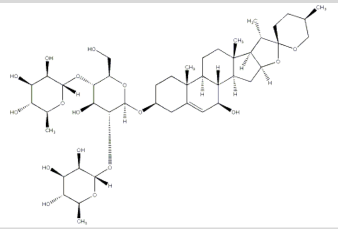
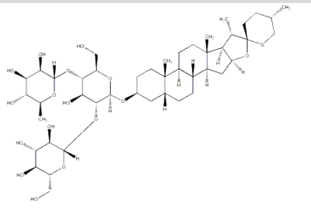
		
C00008838 Epicatechin 3-O-beta-D-allopyranoside $C_{21}H_{24}O_{11}$	C00008841 Catechin 3-O-beta-D-glucopyranoside $C_{21}H_{24}O_{11}$	C00008842 Epicatechin 3-O-beta-D-glucopyranoside $C_{21}H_{24}O_{11}$
		
C00033501 (+)-Catechin 3-O-beta-D-galactopyranoside $C_{21}H_{24}O_{11}$	C00008686 Fustin 3-glucoside $C_{21}H_{22}O_{11}$	C00008703 Astilbin $C_{21}H_{22}O_{11}$
		
C00008704 Isoastibin $C_{21}H_{22}O_{11}$	C00008705 Neoisoastilbin $C_{21}H_{22}O_{11}$	

Figure S1(e): Cluster 5, 14 Elements, Flavanonol

		
C00006356 6-C-Glucopyranosylepicat echin $C_{21}H_{24}O_{11}$	C00006357 8-C-Glucopyranosylepicatechin $C_{21}H_{24}O_{11}$	C00008846 Catechin-5-O-beta-D-glucopyranosi de (+)-Catechin-5-O-beta-D-glucopyranosid e $C_{21}H_{24}O_{11}$
		
C00006359	C00006358	C00008847

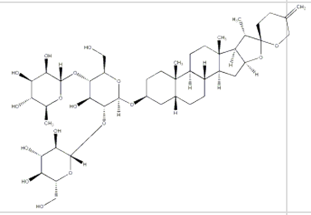
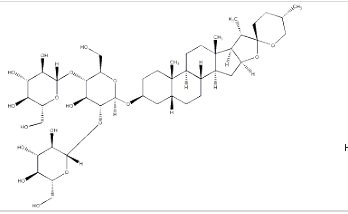
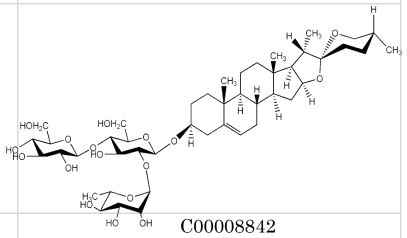
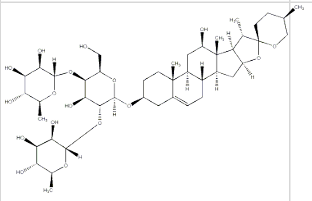
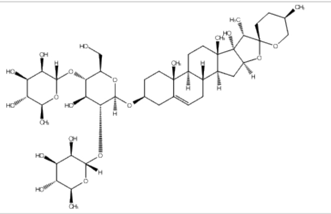
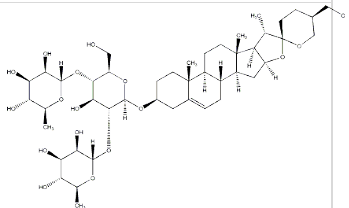
8-C-Glucopyranosylcatec hin C21H24O11	6-C-Glucopyranosylcatechin C21H24O11	Epicatechin-5-O-beta-D-glucopyrano side (-)-Epicatechin-5-O-beta-D-glucopyranos ide C21H24O11
		
C00008838 Epicatechin 3-O-beta-D-allopyranoside C21H24O11	C00008841 Catechin 3-O-beta-D-glucopyranoside C21H24O11	C00008842 Epicatechin 3-O-beta-D-glucopyranoside C21H24O11
		
C00033501 (+)-Catechin 3-O-beta-D-galactopyranoside C21H24O11	C00008686 Fustin 3-glucoside C21H22O11	C00008703 Astilbin C21H22O11

Figure S1(f): Cluster 6, 12 Elements, Saponin

Table 2: miRNA and their total relevance score

miRNA	DIANA	miRecord	mirTarbase	mirWalk	Rscore	Database	TRscore
hsa-let-7e-5p	1.03			27.61	28.64	2	57.29
hsa-miR-1-3p	24.41		0.02		24.42	2	48.85
hsa-let-7b-5p	3.25		0.02	9.14	12.40	3	37.21
hsa-miR-103a-3p	3.37		0.02	8.11	11.50	3	34.50
hsa-let-7d-5p	0.22			16.98	17.20	2	34.39
hsa-let-7a-5p	2.30		0.02	6.62	8.94	3	26.82
hsa-miR-107	3.33			8.37	11.69	2	23.38
hsa-miR-16-5p	11.27		0.25		11.52	2	23.04
hsa-let-7c-5p	0.91			8.11	9.02	2	18.04
hsa-miR-106b-5p	2.24		0.25	2.53	5.02	3	15.06
hsa-miR-15a-5p	7.08		0.25		7.33	2	14.66
hsa-miR-1207-5p				12.14	12.14	1	12.14
hsa-miR-1182				11.77	11.77	1	11.77
hsa-miR-15b-5p	5.31		0.14		5.45	2	10.90
hsa-miR-106a-5p	0.65		0.02	2.85	3.52	3	10.55
hsa-miR-124-3p	4.04			1.22	5.25	2	10.50
hsa-miR-1224-5p				8.73	8.73	1	8.73
hsa-miR-1228-5p				8.56	8.56	1	8.56
hsa-let-7f-5p	2.78			1.10	3.88	2	7.76
hsa-miR-1178-3p				7.65	7.65	1	7.65
hsa-miR-20a-5p	3.28		0.14		3.42	2	6.84
hsa-miR-27a-3p	3.34		0.06		3.41	2	6.81
hsa-let-7g-5p	1.40		0.02	0.76	2.18	3	6.53
hsa-let-7i-5p	1.58			1.62	3.20	2	6.39
hsa-miR-17-5p	2.94		0.14		3.08	2	6.15
hsa-miR-1202				5.86	5.86	1	5.86
hsa-miR-29b-3p	0.61		2.13		2.74	2	5.48
hsa-miR-424-5p	2.55		0.14		2.69	2	5.38
hsa-miR-93-5p	2.56		0.06		2.62	2	5.25
hsa-miR-1236-3p				4.52	4.52	1	4.52
hsa-miR-1229-3p				4.28	4.28	1	4.28
hsa-miR-26a-5p	1.75		0.25		2.00	2	4.01
hsa-miR-98-5p	1.82		0.02		1.84	2	3.67
hsa-miR-16		3.64			3.64	1	3.64
hsa-miR-29c-3p	0.73		0.94		1.67	2	3.34
hsa-miR-27b-3p	1.62		0.02		1.64	2	3.28
hsa-miR-1226-3p				3.22	3.22	1	3.22
hsa-miR-23b-3p	1.50		0.02		1.51	2	3.02
hsa-miR-1234-3p				3.00	3.00	1	3.00
hsa-miR-29a-3p	0.73		0.69		1.42	2	2.84
hsa-miR-221-3p			2.72		2.72	1	2.72
hsa-miR-1228-3p				2.65	2.65	1	2.65
hsa-miR-195-5p	1.22		0.06		1.28	2	2.56
hsa-miR-15a		2.55			2.55	1	2.55

miRNA	DIANA	miRecord	mirTarbase	mirWalk	Rscore	Database	TRscore
hsa-miR-103a-2-5p				2.20	2.20	1	2.20
hsa-let-7g-3p				2.18	2.18	1	2.18
hsa-miR-200c-3p	0.03		1.00		1.03	2	2.06
hsa-let-7e-3p				1.83	1.83	1	1.83
hsa-miR-30a-5p	1.80				1.80	1	1.80
hsa-miR-222-3p			1.80		1.80	1	1.80
hsa-miR-155-5p	0.71		0.14		0.85	2	1.70
hsa-miR-106b-3p				1.62	1.62	1	1.62
hsa-miR-181a-5p	0.62		0.19		0.80	2	1.61
hsa-miR-1225-5p				1.57	1.57	1	1.57
hsa-miR-30d-5p	1.56				1.56	1	1.56
hsa-miR-200b-3p	0.00		0.75		0.75	2	1.50
hsa-miR-10a-5p	0.01			0.75	0.75	2	1.50
hsa-miR-30c-5p	1.46				1.46	1	1.46
hsa-miR-92a-3p	0.71		0.02		0.72	2	1.45
hsa-miR-19a-3p	1.44				1.44	1	1.44
hsa-miR-30e-5p	1.42				1.42	1	1.42
hsa-miR-1207-3p				1.40	1.40	1	1.40
hsa-miR-1180-3p				1.36	1.36	1	1.36
hsa-miR-124-5p				1.33	1.33	1	1.33
hsa-miR-30b-5p	1.20				1.20	1	1.20
hsa-miR-34a-5p	0.53		0.06		0.59	2	1.18
hsa-miR-34c-5p			1.13		1.13	1	1.13
hsa-miR-1180-5p				1.12	1.12	1	1.12
hsa-miR-17		1.09			1.09	1	1.09
hsa-miR-19b-3p	1.08				1.08	1	1.08
hsa-miR-24-3p	0.04		0.47		0.51	2	1.01
hsa-miR-25-3p	0.43		0.02		0.45	2	0.89
hsa-miR-125b-5p	0.04		0.39		0.43	2	0.86
hsa-miR-1181				0.83	0.83	1	0.83
hsa-miR-181b-5p	0.20		0.19		0.39	2	0.78
hsa-miR-130a-3p	0.76				0.76	1	0.76
hsa-miR-148a-3p	0.23		0.14		0.37	2	0.74
hsa-miR-122-5p				0.71	0.71	1	0.71
hsa-miR-21-5p	0.03		0.31		0.34	2	0.69
hsa-miR-429		0.09	0.25		0.34	2	0.68
hsa-miR-29a		0.64			0.64	1	0.64
hsa-miR-34b-3p			0.61		0.61	1	0.61
hsa-miR-23a-3p	0.61				0.61	1	0.61
hsa-miR-1247-5p				0.60	0.60	1	0.60
hsa-miR-301a-3p	0.59				0.59	1	0.59
hsa-miR-29b		0.55			0.55	1	0.55
hsa-miR-181a		0.55			0.55	1	0.55
hsa-miR-148b-3p	0.21		0.06		0.27	2	0.54
hsa-miR-497-5p	0.11		0.14		0.25	2	0.50

miRNA	DIANA	miRecord	mirTarbase	mirWalk	Rscore	Database	TRscore
hsa-miR-130b-3p	0.50				0.50	1	0.50
hsa-miR-10b-5p	0.02			0.21	0.23	2	0.47
hsa-miR-101-5p				0.47	0.47	1	0.47
hsa-miR-301b-3p	0.44				0.44	1	0.44
hsa-miR-7-5p	0.20		0.02		0.22	2	0.44
hsa-miR-221		0.41			0.41	1	0.41
hsa-miR-1		0.41			0.41	1	0.41
hsa-miR-181c-5p	0.17		0.03		0.20	2	0.40
hsa-miR-20b-5p	0.39				0.39	1	0.39
hsa-miR-34b		0.36			0.36	1	0.36
hsa-miR-222		0.36			0.36	1	0.36
hsa-miR-20a		0.36			0.36	1	0.36
hsa-miR-128-3p	0.11		0.06		0.17	2	0.35
hsa-miR-26b-5p	0.35				0.35	1	0.35
hsa-miR-10b-3p				0.34	0.34	1	0.34
hsa-let-7c-3p				0.33	0.33	1	0.33
hsa-miR-199a-5p			0.31		0.31	1	0.31
hsa-let-7b-3p				0.29	0.29	1	0.29
hsa-miR-105-5p				0.28	0.28	1	0.28
hsa-miR-34c		0.27			0.27	1	0.27
hsa-miR-181b		0.27			0.27	1	0.27
hsa-miR-125a-5p	0.04		0.09		0.13	2	0.26
hsa-miR-32-5p	0.25				0.25	1	0.25
hsa-miR-454-3p	0.20				0.20	1	0.20
hsa-miR-503-5p	0.04		0.06		0.10	2	0.20
hsa-miR-152-3p			0.19		0.19	1	0.19
hsa-miR-372		0.18			0.18	1	0.18
hsa-miR-19a		0.18			0.18	1	0.18
hsa-miR-181d		0.18			0.18	1	0.18
hsa-miR-181c		0.18			0.18	1	0.18
has-miR-93		0.18			0.18	1	0.18
has-miR-16		0.18			0.18	1	0.18
has-miR-15a		0.18			0.18	1	0.18
hsa-miR-29c		0.14			0.14	1	0.14
hsa-miR-92b-3p	0.13				0.13	1	0.13
hsa-miR-181d-5p	0.12				0.12	1	0.12
hsa-miR-101-3p	0.04			0.01	0.05	2	0.11
hsa-miR-1245a				0.10	0.10	1	0.10
hsa-miR-302d-3p			0.09		0.09	1	0.09
hsa-miR-17-3p			0.09		0.09	1	0.09
hsa-miR-93		0.09			0.09	1	0.09
hsa-miR-34a		0.09			0.09	1	0.09
hsa-miR-302d		0.09			0.09	1	0.09
hsa-miR-296-5p		0.09			0.09	1	0.09
hsa-miR-200c		0.09			0.09	1	0.09
hsa-miR-200b		0.09			0.09	1	0.09

miRNA	DIANA	miRecord	mirTarbase	mirWalk	Rscore	Database	TRscore
hsa-miR-106b		0.09			0.09	1	0.09
has-let-7a3b		0.09			0.09	1	0.09
has-let-71f1		0.09			0.09	1	0.09
hsa-miR-18a-5p	0.02		0.02		0.04	2	0.08
hsa-miR-520b			0.08		0.08	1	0.08
hsa-miR-425-5p	0.01		0.03		0.04	2	0.07
hsa-miR-1246				0.07	0.07	1	0.07
hsa-miR-520a-3p			0.06		0.06	1	0.06
hsa-miR-372-3p			0.06		0.06	1	0.06
hsa-miR-208b-3p			0.06		0.06	1	0.06
hsa-miR-208a-3p			0.06		0.06	1	0.06
hsa-miR-203a-3p			0.06		0.06	1	0.06
hsa-miR-200a-3p			0.06		0.06	1	0.06
hsa-miR-141-3p			0.06		0.06	1	0.06
hsa-miR-133b			0.06		0.06	1	0.06
hsa-miR-133a-3p			0.06		0.06	1	0.06
hsa-miR-126-3p			0.06		0.06	1	0.06
hsa-miR-199a-3p	0.01		0.02		0.02	2	0.05
hsa-miR-182-5p	0.01		0.02		0.02	2	0.05
hsa-miR-99a		0.05			0.05	1	0.05
hsa-miR-503		0.05			0.05	1	0.05
hsa-miR-424		0.05			0.05	1	0.05
hsa-miR-302c		0.05			0.05	1	0.05
hsa-miR-302a		0.05			0.05	1	0.05
hsa-miR-21		0.05			0.05	1	0.05
hsa-miR-200a		0.05			0.05	1	0.05
hsa-miR-15b		0.05			0.05	1	0.05
hsa-miR-155		0.05			0.05	1	0.05
hsa-miR-153		0.05			0.05	1	0.05
hsa-miR-141		0.05			0.05	1	0.05
hsa-miR-125b		0.05			0.05	1	0.05
hsa-miR-106a		0.05			0.05	1	0.05
hsa-miR-100		0.05			0.05	1	0.05
hsa-let-7b		0.05			0.05	1	0.05
has-miR-17-19b-1		0.05			0.05	1	0.05
has-miR-16-1		0.05			0.05	1	0.05
hsa-miR-100-5p	0.02			0.00	0.02	2	0.04
hsa-miR-342-3p	0.00		0.02		0.02	2	0.03
hsa-miR-185-5p	0.00		0.02		0.02	2	0.03
hsa-miR-296-3p			0.03		0.03	1	0.03
hsa-miR-365b-3p	0.03				0.03	1	0.03
hsa-miR-365a-3p	0.03				0.03	1	0.03
hsa-miR-100-3p				0.02	0.02	1	0.02
hsa-miR-122-3p				0.02	0.02	1	0.02
hsa-miR-548o-5p	0.02				0.02	1	0.02
hsa-miR-548c-5p	0.02				0.02	1	0.02

miRNA	DIANA	miRecord	mirTarbase	mirWalk	Rscore	Database	TRscore
hsa-miR-548am-5p	0.02				0.02	1	0.02
hsa-miR-30e-3p	0.02				0.02	1	0.02
hsa-miR-543			0.02		0.02	1	0.02
hsa-miR-524-5p			0.02		0.02	1	0.02
hsa-miR-494-3p			0.02		0.02	1	0.02
hsa-miR-451a			0.02		0.02	1	0.02
hsa-miR-449a			0.02		0.02	1	0.02
hsa-miR-448			0.02		0.02	1	0.02
hsa-miR-409-3p			0.02		0.02	1	0.02
hsa-miR-381-3p			0.02		0.02	1	0.02
hsa-miR-335-3p			0.02		0.02	1	0.02
hsa-miR-330-5p			0.02		0.02	1	0.02
hsa-miR-326			0.02		0.02	1	0.02
hsa-miR-324-5p			0.02		0.02	1	0.02
hsa-miR-31-5p			0.02		0.02	1	0.02
hsa-miR-214-3p			0.02		0.02	1	0.02
hsa-miR-211-5p			0.02		0.02	1	0.02
hsa-miR-205-5p			0.02		0.02	1	0.02
hsa-miR-204-5p			0.02		0.02	1	0.02
hsa-miR-202-3p			0.02		0.02	1	0.02
hsa-miR-197-3p			0.02		0.02	1	0.02
hsa-miR-193b-3p			0.02		0.02	1	0.02
hsa-miR-193a-3p			0.02		0.02	1	0.02
hsa-miR-1908-5p			0.02		0.02	1	0.02
hsa-miR-1826			0.02		0.02	1	0.02
hsa-miR-149-5p			0.02		0.02	1	0.02
hsa-miR-145-5p			0.02		0.02	1	0.02
hsa-miR-143-3p			0.02		0.02	1	0.02
hsa-miR-141-5p			0.02		0.02	1	0.02
hsa-miR-140-3p			0.02		0.02	1	0.02
hsa-miR-135b-5p			0.02		0.02	1	0.02
hsa-miR-132-3p			0.02		0.02	1	0.02
hsa-miR-126-5p			0.02		0.02	1	0.02
hsa-miR-18b-5p	0.02				0.02	1	0.02
hsa-miR-671-5p	0.01				0.01	1	0.01
hsa-miR-320b	0.01				0.01	1	0.01
hsa-miR-320a	0.01				0.01	1	0.01
hsa-miR-502-3p	0.01				0.01	1	0.01
hsa-miR-501-3p	0.01				0.01	1	0.01
hsa-miR-374b-5p	0.01				0.01	1	0.01
hsa-miR-320d	0.01				0.01	1	0.01
hsa-miR-22-3p	0.01				0.01	1	0.01
hsa-miR-19b-1-5p	0.01				0.01	1	0.01
hsa-miR-629-5p	0.01				0.01	1	0.01
hsa-miR-548j-5p	0.01				0.01	1	0.01

miRNA	DIANA	miRecord	mirTarbase	mirWalk	Rscore	Database	TRscore
hsa-miR-423-5p	0.01				0.01	1	0.01
hsa-miR-19a-5p	0.01				0.01	1	0.01
hsa-miR-199b-3p	0.01				0.01	1	0.01
hsa-miR-196a-5p	0.01				0.01	1	0.01
hsa-miR-10a-3p				0.01	0.01	1	0.01
hsa-miR-30d-3p	0.00				0.00	1	0.00
hsa-miR-30a-3p	0.00				0.00	1	0.00
hsa-miR-302a-3p	0.00				0.00	1	0.00
hsa-miR-96-5p	0.00				0.00	1	0.00
hsa-miR-9-5p	0.00				0.00	1	0.00
hsa-miR-548z	0.00				0.00	1	0.00
hsa-miR-548h-3p	0.00				0.00	1	0.00
hsa-miR-501-5p	0.00				0.00	1	0.00
hsa-miR-500b-5p	0.00				0.00	1	0.00
hsa-miR-374a-5p	0.00				0.00	1	0.00
hsa-miR-186-5p	0.00				0.00	1	0.00
hsa-miR-153-3p	0.00				0.00	1	0.00
hsa-miR-99a-5p	0.00				0.00	1	0.00
hsa-miR-629-3p	0.00				0.00	1	0.00
hsa-miR-576-3p	0.00				0.00	1	0.00
hsa-miR-574-5p	0.00				0.00	1	0.00
hsa-miR-548d-5p	0.00				0.00	1	0.00
hsa-miR-522-5p	0.00				0.00	1	0.00
hsa-miR-449c-5p	0.00				0.00	1	0.00
hsa-miR-367-3p	0.00				0.00	1	0.00
hsa-miR-363-3p	0.00				0.00	1	0.00
hsa-miR-362-5p	0.00				0.00	1	0.00
hsa-miR-34b-5p	0.00				0.00	1	0.00
hsa-miR-33b-5p	0.00				0.00	1	0.00
hsa-miR-33a-5p	0.00				0.00	1	0.00
hsa-miR-320c	0.00				0.00	1	0.00
hsa-miR-302c-3p	0.00				0.00	1	0.00
hsa-miR-218-5p	0.00				0.00	1	0.00
hsa-miR-196b-5p	0.00				0.00	1	0.00
hsa-miR-146b-5p	0.00				0.00	1	0.00
hsa-miR-146a-5p	0.00				0.00	1	0.00
hsa-miR-142-5p	0.00				0.00	1	0.00
hsa-miR-142-3p	0.00				0.00	1	0.00
hsa-miR-137	0.00				0.00	1	0.00

Java Code

```
/*
 * DPClusSBO1.1 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * DPClusSBO1.1 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */

package com.bb.graph;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Scanner;
import java.util.Set;
import javax.swing.JOptionPane;
import org.apache.poi.openxml4j.opc.OPCPackage;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
```

This is the class to convert a sparse matrix to the simple graph format using Tanimoto coefficient and relation number. The main class for BiClusO

```
public class clsMatrixData {
    private String[] rname ;
    private String[] cname;
    private int[][]  data;
    private int[][]  rl;
    private float[][]  tf;

    private int rsize;
    private int csize;

    private int thrl;
    private double thtf;
    private int rdirection;

    private String fpath="";
    private String fsep ="";
    private ArrayList<String> bindata;

    public int getmatdata(String ft, String st)
    {
     int i =Arrays.asList(rname).indexOf(ft);
     int j =Arrays.asList(cname).indexOf(st);

     return data[i][j];
```

```
}

public ArrayList getclssecondtuple(ArrayList clsn , double pr )
{
 ArrayList<String> st= new ArrayList<>();
 double sum, pp;
 for (int j= 0 ; j<= csize-1; ++j)
 {
 sum = 0;
  for(int i = 0; i<=clsn.size()-1; ++i)
  {
   String  sp = (String) clsn.get(i);
    int p = Arrays.asList(rname).indexOf(sp);

     if (data[p][j]==1) sum = sum+1;
  }
  pp = (double) sum/clsn.size();
  if (pp >= pr) st.add((String) cname[j]);

 }
 return st;
}

public ArrayList removeisolatenode(ArrayList st, ArrayList ft )
{
 ArrayList<String> nt= new ArrayList<>();
 String sp1, sp2;
 int p,q, flag,j;
 for (int i= 0 ; i<= st.size()-1; ++i)
 {
    sp1 = (String) st.get(i);
    p = Arrays.asList(rname).indexOf(sp1);
    flag=0;
    j=0;
    while ((flag==0) && (j<ft.size()))
    {
     sp2 = (String) ft.get(j);
     q = Arrays.asList(cname).indexOf(sp2);
     if (data[p][q]==1) flag=1;
     j= j+1;
    }
   if (flag==1) nt.add(sp1);
 }
  return nt;
}

public clsMatrixData(String fp , String sep, int rl , double tf) {
    fpath = fp;
    fsep = sep;
    thrl = rl;
    thtf = tf;
    rdirection = 0;
}

public clsMatrixData(String fp , String sep, int rl , double tf, int rd) {
    fpath = fp;
    fsep = sep;
    thrl = rl;
    thtf = tf;
    rdirection = rd;
}


public clsMatrixData(String fp, String sep ) {
      fpath = fp;
      fsep = sep;
      thrl = 0;
      thtf = 0.0;
  }
  public ArrayList<String> getBindata() {
      return bindata;
  }

  public int getThrl() {
      return thrl;
  }

  public double getThtf() {
      return thtf;
  }

  public void processdata()
  {
     if (rdirection==0) datareadexcel();
     else datareadexcelrev();
     relation();
     edgeset();
  }

  public void processdatatxt()
  {
     if (rdirection==0) datareadcsv(0);
     else datareadcsv(1);
     relation();
     edgeset();
  }


  public void calculatethresold()
  {
    // dataread();
     relation();
     findthresold();
     findthresoldrl();

  }


public void findthresoldrl()
{
 ArrayList<String> edge = new ArrayList <>();
 int edgenum[] ;
 int k=0;
 int maxrl=0 , minrl= 999999999;
  for (int i= 0; i<= rsize-1; ++i)
   {
    for (int j= 0; j<= i-1; ++j)
      {
         if (maxrl < rl[i][j]) maxrl = rl[i][j];
         if ((minrl > rl[i][j]) && (rl[i][j]>0)) minrl = rl[i][j];

      }
   }
 // System.out.println("Printing max min rl " + maxrl + "  " +minrl);
 edgenum = new int[maxrl-minrl+1];

for (int t = minrl; t<= maxrl; ++t)
 {
   edge = new ArrayList <>();
   for (int i= 0; i<= rsize-1; ++i)
   {
   for (int j= 0; j<= i-1; ++j)
    {
     if ((rl[i][j] >= t))
      {
        if  (!edge.contains(rname[i])) edge.add(rname[i]);
        if  (!edge.contains(rname[j])) edge.add(rname[j]);
      }
     }
   }
   edgenum[k] = edge.size();
   ++k;
 }

//    System.out.println("Printing thresold value   ");
// for(int i=0; i<=k-1; ++i)
//     System.out.print(edgenum[i] + "    ");


 int totedge = 0;
 for(int i=k-1; i>=0; --i)
 {
    if (totedge < edgenum[i])
    {
      totedge = edgenum[i];
       thrl = maxrl-i;
    }
 }
//   System.out.print("thresold" + " " + thrl);

}



 public void relation()
 {
 int rnum , cnum;
  for (int i= 0; i<= rsize-1; ++i)
   {
   for (int j= 0; j<= i-1; ++j)
     {
       rnum = 0;
       cnum = 0 ;
       for (int k=0 ; k<= csize-1; ++k )
        {
          if ((data[i][k]==1) && (data[j][k]==1))
          {
           rnum = rnum +1;
          }
          if ((data[i][k] + data[j][k])==1)
          {
           cnum = cnum +1;
          }

        }
       rl[i][j] = rnum;
       tf[i][j] = ((float) rnum) / ((float) rnum + (float) cnum) ;
       //tf[i][j] = (float) cnum ;

     }
   }
```

```
}


public void datareadexcel()
{
  int i=0, j=0;
 try {
     FileInputStream inputStream = new FileInputStream(new File(fpath));
     //FileInputStream iS = new FileInputStream(f);
      File f = new File(fpath);
      OPCPackage opcPackage = OPCPackage.open(f.getAbsolutePath());
       XSSFWorkbook  workbook = new XSSFWorkbook(opcPackage);

     // XSSFWorkbook  workbook = new XSSFWorkbook(inputStream);
      //  Workbook workbook = WorkbookFactory.create(inputStream);

       Sheet firstSheet = (Sheet) workbook.getSheetAt(0);
       Iterator<Row> iterator = firstSheet.iterator();
        rsize =  firstSheet.getLastRowNum();
        rname = new String[rsize];
        csize = firstSheet.getRow(0).getLastCellNum()-1;
        cname = new String[csize];
        data = new int[rsize][csize];
       // clsWindow.matdata = new int[rsize][csize];
        rl = new int[rsize][rsize];
        tf = new float[rsize][rsize];

      for (i=1; i<= csize;  ++i)
      {
       (firstSheet.getRow(0).getCell(i)).setCellType(Cell.CELL_TYPE_STRING);
        cname[i-1]=   (firstSheet.getRow(0).getCell(i).toString()).trim();
      }
      for (i=1; i<= rsize; ++i)
      {
       rname[i-1]=   (firstSheet.getRow(i).getCell(0).toString()).trim();
      }

     String ss;
     double d=0;
     for (i=1; i<= rsize; ++i)
       {
        for (j=1; j<= csize; ++j)
        {
           ss  = (firstSheet.getRow(i).getCell(j).toString()) ;
           d = Double.parseDouble(ss);
            data[i-1][j-1] = (int) d ;
      //     clsWindow.matdata[i-1][j-1]= (int) d ;
         }
       }

 //    System.out.println("end");
      i= i+2;

 } catch (Exception e) {
    JOptionPane.showMessageDialog(null, e.toString());
 }

}


public void datareadexcelrev()
{
  int i=0, j=0;
 try {
     FileInputStream inputStream = new FileInputStream(new File(fpath));
      //FileInputStream iS = new FileInputStream(f);
       File f = new File(fpath);
       OPCPackage opcPackage = OPCPackage.open(f.getAbsolutePath());
        XSSFWorkbook  workbook = new XSSFWorkbook(opcPackage);

      // XSSFWorkbook  workbook = new XSSFWorkbook(inputStream);
       //  Workbook workbook = WorkbookFactory.create(inputStream);

        Sheet firstSheet = (Sheet) workbook.getSheetAt(0);
        Iterator<Row> iterator = firstSheet.iterator();
         rsize = firstSheet.getRow(0).getLastCellNum()-1;
         rname = new String[rsize];
         csize = firstSheet.getLastRowNum();
         cname = new String[csize];
         data = new int[rsize][csize];
         rl = new int[rsize][rsize];
         tf = new float[rsize][rsize];

       for (i=1; i<= rsize; ++i)
       {
        (firstSheet.getRow(0).getCell(i)).setCellType(Cell.CELL_TYPE_STRING);
         rname[i-1]=   (firstSheet.getRow(0).getCell(i).toString()).trim();
       }

      for (i=1; i<= csize; ++i)
       {
        cname[i-1]=   (firstSheet.getRow(i).getCell(0).toString()).trim();
       }
      String ss;
      double d=0;
      for (i=1; i<= rsize; ++i)
        {
         for (j=1; j<= csize; ++j)
         {
            ss  = (firstSheet.getRow(j).getCell(i).toString()) ;
            d = Double.parseDouble(ss);
             data[i-1][j-1] = (int) d ;
         }

        }

      System.out.println("end");
       i= i+2;

 } catch (Exception e) {
     System.err.println(e);
 }

}

public void datareadcsv(int dir)
{
  String curLine, key ;
  String[] splitted;
   int i=0, j=0;

  Set <String> fnode, snode;
  ArrayList<String> fnodea, snodea;
  HashMap<String, Integer> sdata;

  sdata = new HashMap<>();
  fnode = new HashSet<>();
  snode = new HashSet<>();
```

```
    try {

    Scanner scan = new Scanner(new FileInputStream(fpath));
    i=1;
    while(scan.hasNextLine()){
     curLine = scan.nextLine();
     splitted = curLine.split(fsep);
     if (dir==0){
     fnode.add(splitted[0]);
     snode.add(splitted[1]);
     sdata.put(splitted[0]+"-"+splitted[1],j);
     }
     else{
     fnode.add(splitted[1]);
     snode.add(splitted[0]);
     sdata.put(splitted[1]+"-"+splitted[0],j);
     }
     i++;
    }
     fnodea = new ArrayList<>(fnode);
     snodea = new ArrayList<>(snode);
     rsize = fnodea.size();
     rname = new String[rsize];
     csize = snodea.size();
     cname = new String[csize];
     data = new int[rsize][csize];
     rl = new int[rsize][rsize];
     tf = new float[rsize][rsize];

    for (i = 0; i < fnodea.size(); i++) {
         rname[i]=fnodea.get(i);

       }
    for (i = 0; i < snodea.size(); i++) {
         cname[i]=snodea.get(i);

       }

    for (i = 0; i < fnodea.size(); i++) {
       for (j = 0; j < snodea.size(); j++) {
         key = fnodea.get(i)+"-"+snodea.get(j);
         if (sdata.containsKey(key)) data[i][j]=1;
         else  data[i][j]=0;
        //System.out.print(data[i][j]);
       }
       // System.out.println();

    }

      } catch (Exception e) { System.out.println(e);}
     }

  }
```

```
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */
```

This class is used to find overlapping between biclusters, filter bicluster, Join two bicluster set

```
package com.bb.graph;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Random;
import java.util.Set;
import java.util.Vector;

/**
 *
 * @author hira
 */
public class clsBicluster implements  Serializable{

    private HashMap<Integer, clsBinode> bclus;

     ArrayList<String> cft = new ArrayList<>();
     ArrayList<String> cst = new ArrayList<>();



     public void setdata()
     {
        ArrayList<String> ft , st;
        clsBinode bn;
        int p=0, q=0;
       for (Integer key : bclus.keySet()) {
        bn = (clsBinode) bclus.get(key);
        ft = bn.getFtuple();
        for(int i=0; i<=ft.size()-1;++i)
        {
         if (!cft.contains(ft.get(i)))
          {
            cft.add(p,ft.get(i));
            ++p;
          }
        }
        st = bn.getStuple();
        for(int i=0; i<=st.size()-1;++i)
        {
         if (!cst.contains(st.get(i)))
          {
            cst.add(q,st.get(i));
            ++q;
          }
        }
      }

     }

    public HashMap<Integer, clsBinode> getBclus() {
        return bclus;
    }
 public clsBicluster() {
    this.bclus = new HashMap<>();
 }

public void clsStcommon()
  {
  clsBinode bn, cn;
  ArrayList<String> std1 ;
  ArrayList<String> std2 ;

  int i ,j,p;
  ArrayList<Integer> keys;
     keys = new ArrayList<Integer>(bclus.keySet());

  for(i=0; i<=keys.size()-1; ++i )
  {
     std1  = new ArrayList<String>(((clsBinode) bclus.get(keys.get(i))).getStuple());

   for(j=0; j<=i-1; ++j )
   {
      std2  =  new ArrayList<String>(((clsBinode) bclus.get(keys.get(j))).getStuple());
      std1.retainAll(std2);
     int com = std1.size();

      std1  = new ArrayList<String>(((clsBinode) bclus.get(keys.get(i))).getStuple());
    //  int ss = std1.size() + std2.size() ;
    //  System.out.print(ss+" ");
    double per;
     per = ((double) com /(double) (std1.size()+std2.size()-com)) *100 ;
  //   int kk = per
  //  System.out.print(per + "-");

      String ss = Integer.toString(i+1) + "," + Integer.toString(j+1);

    //   System.out.print(ss+" ");
   }
    //  System.out.println("\n");

  }


}

 public clsBicluster( Vector msc , clsMatrixData md , double psc ) {
    Vector ss = msc;
    Vector cs;
    ArrayList<String> fs , st , stc, fsn, fsnc;
    clsBinode bn ;
    bclus =  new HashMap<>();
    for(int i=0; i<= ss.size()-1; i++){
       cs=(Vector)ss.elementAt(i);
      fs = new ArrayList<String>(cs);
      st = md.getclssecondtuple(fs,psc);
      fsn = md.removeisolatenode(fs,st);
      stc = new ArrayList<String>(st);
      fsnc = new ArrayList<String>(fsn);

      bn = new clsBinode(fsnc , stc);
     // bn = new clsBinode(fs , stc);
      bclus.put(i+1, bn);
   }

  bclus = sortByValue(bclus);
 }

 public clsBicluster(clsBicluster m1 , clsBicluster m2)
 {
    bclus =  new HashMap<>();
    HashMap<Integer, clsBinode> hs;
    clsBinode bn;
    hs=m1.getBclus();
    int i=0;
    for (Integer key : hs.keySet()) {
     bn = (clsBinode) hs.get(key);
     if ((bn.getFtuple().size()*bn.getStuple().size())>0)
     {
      bclus.put(i+1, bn);
      ++i;
     }
    }
    hs=m2.getBclus();
    for (Integer key : hs.keySet()) {
     bn = new clsBinode(hs.get(key).getStuple() , hs.get(key).getFtuple());
    // bn = (clsBinode) hs.get(key);
    if ((bn.getFtuple().size()*bn.getStuple().size())>0)
     {
      bclus.put(i+1, bn);
      ++i;
     }
    }

  bclus = sortByValue(bclus);
    overlapping(1);
 }

public void overlappinguser(double ov)
{
 overlapping(ov);
 bclus = sortByValue(bclus);
}


private  HashMap<Integer, clsBinode> sortByValue(HashMap<Integer, clsBinode> unsortMap) {
     int i;
     List<HashMap.Entry<Integer, clsBinode>> list =
             new LinkedList<HashMap.Entry<Integer, clsBinode>>(unsortMap.entrySet());

     Collections.sort(list, new Comparator<HashMap.Entry<Integer, clsBinode>>() {
         public int compare(HashMap.Entry<Integer, clsBinode> o1,
                            HashMap.Entry<Integer, clsBinode> o2) {
             Integer i = o1.getValue().getFtuple().size()*o1.getValue().getStuple().size();
             Integer j = o2.getValue().getFtuple().size()*o2.getValue().getStuple().size();

             return j.compareTo(i);
         }
     });

     HashMap<Integer, clsBinode> sortedMap = new LinkedHashMap<Integer, clsBinode>();
     i=1;
     for (HashMap.Entry<Integer, clsBinode> entry : list) {
         sortedMap.put(i, entry.getValue());
         ++i;
     }

     return sortedMap;
 }
```

```
  private void overlapping( double ovlpcf )
   {
    int cl = bclus.size();
    double x1,x2, y1, y2, xy1, xy2;
    double ol=0;
    ArrayList<String> xx1, xx2 = new ArrayList<>();
    ArrayList<String> yy1, yy2 = new ArrayList<>();
    ArrayList<String> xx, yy = new ArrayList<>();

    ArrayList<Integer> ritem = new ArrayList<>();
    double [][] ovlp = new double[cl][cl];
    int[] kset  ;
    kset = new int[cl];
    int index=0;
    for(Integer element : bclus.keySet())
        kset[index++] = element.intValue();

     for(int i=2; i<= cl; i++){
      xx1=(ArrayList)bclus.get(kset[i-1]).getFtuple();
      yy1=(ArrayList)bclus.get(kset[i-1]).getStuple();

      for(int j=1; j<= i-1; j++){
        xx2=(ArrayList)bclus.get(kset[j-1]).getFtuple();
        yy2=(ArrayList)bclus.get(kset[j-1]).getStuple();
        x1  = xx1.size();
        x2  = xx2.size();
        y1  = yy1.size();
        y2  = yy2.size();
        xx = new ArrayList<String>();
        xx.addAll(xx1);
        xx.retainAll(xx2);
        yy = new ArrayList<String>();
        yy.addAll(yy1);
        yy.retainAll(yy2);

        xy1 = xx.size();
        xy2 = yy.size();
        //ovlp[i-1][j-1]=(x2*y2)/(x1*y1 + x2*y2 - xy1*xy2);
        ovlp[i-1][j-1]=(xy1*xy2)/(x1*y1 + x2*y2 - xy1*xy2);

        //ovlp[i-1][j-1]=xy1*xy2/(x2*y2);

       //  System.out.print(String.format("%.2f", ovlp[i-1][j-1]));
       //  System.out.print(" , ");

      }
    //  System.out.println();

    }

    //Vector ritem = new Vector();
    for(int i=2; i<= cl; i++){
       for(int j=1; j<= i-1; j++){
         if(ovlp[i-1][j-1] >= ovlpcf){
             ritem.add(kset[i-1]);
        }
     }
     }
   bclus.keySet().removeAll(ritem);

}

public void clusterjoin(double ovlpcf)
{
 boolean ovchk = true;
 int cl , mi=0, mj=0,index;
 double  ovlp ,maxov ;
 clsBinode bn ;

 ArrayList<String> xx1, xx2 = new ArrayList<>();
 ArrayList<String> yy1, yy2 = new ArrayList<>();
 ArrayList<String> xx, yy = new ArrayList<>();
 Set<String> nxx, nyy = new HashSet<>();

 double x1,x2, y1, y2, xy1, xy2;
 int[] kset  ;
 while (ovchk)
 {
   cl = bclus.size();
//   ovlp = new double[cl][cl];
   kset = new int[cl];
   index=0;
   for(Integer element : bclus.keySet())
       kset[index++] = element.intValue();

   maxov = 0;
   for(int i=1; i<= cl-1; i++){
     xx1=(ArrayList)bclus.get(kset[i]).getFtuple();
     yy1=(ArrayList)bclus.get(kset[i]).getStuple();
     for(int j=0; j<= i-1; j++){
       xx2=(ArrayList)bclus.get(kset[j]).getFtuple();
       yy2=(ArrayList)bclus.get(kset[j]).getStuple();
       x1  = xx1.size();
       x2  = xx2.size();
       y1  = yy1.size();
       y2  = yy2.size();
       xx = new ArrayList<String>();
       xx.addAll(xx1);
       xx.retainAll(xx2);
       yy = new ArrayList<String>();
       yy.addAll(yy1);
       yy.retainAll(yy2);
       xy1 = xx.size();
       xy2 = yy.size();
       //ovlp[i-1][j-1]=(x2*y2)/(x1*y1 + x2*y2 - xy1*xy2);
       ovlp=(xy1*xy2)/(x1*y1 + x2*y2 - xy1*xy2);
       if (ovlp> maxov ){
            maxov = ovlp;
            mi = kset[i];
            mj = kset[j];
            }
       }
     }
     if (maxov > ovlpcf){
       xx1=(ArrayList)bclus.get(mj).getFtuple();
       yy1=(ArrayList)bclus.get(mj).getStuple();
       xx2=(ArrayList)bclus.get(mi).getFtuple();
       yy2=(ArrayList)bclus.get(mi).getStuple();
       nxx = new HashSet<>(xx1);
       nxx.addAll(xx2);
       nyy = new HashSet<>(yy1);
       nyy.addAll(yy2);
       xx1 = new ArrayList<>(nxx);
       yy1 = new ArrayList<>(nyy);
       bn = new clsBinode(xx1, yy1);
       bclus.put(mj, bn);
       bclus.remove(mi);
```

```
      }
     else
      {
       ovchk=false;
      }
  }



}

public void filtercluster(int from, int to)
{
ArrayList<Integer> ritem = new ArrayList<>();
int csize;
for (Integer key : bclus.keySet()) {
  csize = (bclus.get(key).getFtuple().size())*(bclus.get(key).getStuple().size());
  if ((csize < from)  || (csize > to))
  {
   ritem.add(key);
  }
 }
 bclus.keySet().removeAll(ritem);
}

public void filterclustersel(ArrayList sl)
{
 bclus.keySet().removeAll(sl);

}

}

/*
 * DPClusSBO1.1 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * DPClusSBO1.1 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */
```

DpClusO algorithm is implemented in this class. It also sets the overlapping parameter of simple cluster and set the coordinate of cluster set according to different layout

```
package com.bb.graph;

import java.awt.Color;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Objects;
import java.util.StringTokenizer;
import java.util.Vector;
import java.util.concurrent.ThreadLocalRandom;
import javax.swing.JOptionPane;

public class myCluster  implements  Serializable{
 graph maingraph;
 Vector cluster = new Vector() ;
 ArrayList<Integer> clusterno = new ArrayList<>();

 ArrayList<region> frmregion = new ArrayList<region>();

private  Vector AllClusters = new Vector();
private  graph originalGraph;
private  graph inputGraph  ;

public  void setclusterno ()
{
   clusterno.clear();
  for(int i=0; i<= cluster.size()-1; i++){
         clusterno.add(i+1);
       }
}

public  void filterclustersel(ArrayList sl)
{
 Vector ritem = new Vector();

 for (int i=0; i<=clusterno.size()-1; ++i)
  {
   // if (sl.contains(i+1)==true)
    if (sl.contains(clusterno.get(i))==true)

      {
        ritem.add(cluster.elementAt(i));
    }
  }
  int j =sl.size()-1 ;
  for(int i = clusterno.size()-1; i>=0 && j>=0   ; i-- )
  {
    if (Objects.equals(clusterno.get(i), sl.get(j)))
    { clusterno.remove(i);
      --j;
     }
  }

  cluster.removeAll(ritem);


}


public  void filterclustercmb(int fsize , int tsize )
{
 cluster.sort(new MyCompare());
 Vector ritem = new Vector();
 Vector cls = new Vector();
 ArrayList sl = new ArrayList();

 for (int i=0; i<=cluster.size()-1; ++i)
 {
  cls = (Vector) cluster.elementAt(i);
  if ((cls.size() < fsize) | | (cls.size() > tsize))
  {
      ritem.add(cluster.elementAt(i));
      sl.add(i);
  }
 }
```

```
    cluster.removeAll(ritem);
   int j =sl.size()-1 ;
   for(int i = clusterno.size()-1; i>=0 && j>=0   ; i-- )
   {
     if (Objects.equals(i, sl.get(j)))
     { clusterno.remove(i);
       --j;
      }
   }


 }


public  void clusterjoin(double ovlpcf)
{
    cluster.sort(new MyCompare());
    boolean cflg= true;
    int cl = cluster.size();
    Vector  xx;
    Vector yy , ds;
    Vector amal = new Vector();
    double ol=0 , max =0;
    int mi=-1, mj =-1;
    Vector temp = new Vector();
    double [][] ovlp = new double[cl][cl];

    while (cflg)
    {
        max = 0;
        mi = -1;
        mj = -1;
        for(int i=0; i<= cluster.size()-1; i++)
          {
          xx=(Vector)cluster.elementAt(i);
          for(int j=0; j<= i-1; j++)
             {
               yy=(Vector)cluster.elementAt(j);
               ol= overlapP(xx, yy);
                if (max < ol )
                 {
                   max = ol;
                   mi = i;
                   mj = j;
                 }

             }
          }
        if (max >= ovlpcf)
        {
          xx=(Vector)cluster.elementAt(mi);
          yy=(Vector)cluster.elementAt(mj);
          ds = new Vector();
          ds.addAll(xx);
          for(int i=0; i<= yy.size()-1; ++i)
             if (!ds.contains(yy.elementAt(i)))
             {
              ds.add(yy.elementAt(i));
             }

          cluster.remove(mi);
          cluster.remove(mj);
         amal.add(ds);
        }
        else
         {
          cflg = false;
         }
     }

    for(int i =0 ; i<= amal.size()-1; ++i )
    {
      cluster.add(amal.elementAt(i));
    }
   cluster.sort(new MyCompare());

}



public  void filtercluster( double ovlpcf )
 {

     cluster.sort(new MyCompare());
     Vector  xx;
     Vector yy;
     int cl = cluster.size();
     double ol=0;
     double [][] ovlp = new double[cl][cl];

     for(int i=0; i<= cl-1; i++){
       xx=(Vector)cluster.elementAt(i);
       for(int j=0; j<= cl-1; j++){
         yy=(Vector)cluster.elementAt(j);
         ol= overlapP(xx, yy);
         ovlp[i][j]=ol;
       }
     }

     Vector ritem = new Vector();
     for(int i=1; i<= cl-1; i++){
        for(int j=0; j<= i-1; j++){
          if(ovlp[i][j] > ovlpcf){

              ritem.add(cluster.elementAt(i));

        }
      }
      }
     cluster.removeAll(ritem);
      setclusterno();
 }

 public  double overlapP(Vector A, Vector B){

     int lengthA = A.size();
     int lengthB = B.size();
     int count = 0;
     double ratio = 0.0;
     //System.out.println(lengthA+"\t"+lengthB);
     String [] AAry = new String[lengthA];
     String [] BAry = new String[lengthB];
       for(int i=0; i<= lengthA-1; i++)
         AAry[i] = (String)(A.elementAt(i));
       for(int i=0; i<= lengthB-1; i++)
         BAry[i] = (String)(B.elementAt(i));

       for(int i=0; i<= lengthA-1; i++){
        for(int j=0; j<= lengthB-1; j++){
          if(AAry[i].equals(BAry[j])){
             count=count+1;
             break;
             }
            }
      }

    ratio = (double)count*count/(lengthA*lengthB);
    return ratio;

}




public   void setclustercoor(int wxcor , int wycor)
{
  int csize=0 ,x1=0,y1=0,x2=0,y2=0;
  double xs=0 , ys =0;
  int xpw=0, ypw = 0;
  int xrmd=0 , yrmd = 0;
 // csize = cluster.size();
  //csize = 35;
  Vector cs;
   for(int i=0; i<= cluster.size()-1; i++){
        cs=(Vector)cluster.elementAt(i);
       // if(cs.size() >= 3){ csize = csize+1;  }
   csize = csize+1;
   }

  xs = Math.sqrt(4*csize/3);
  ys = Math.sqrt(3*csize/4);
  //xs = Math.sqrt(6*csize/3);
  //ys = Math.sqrt(3*csize/5);

  if ((xs % 1) >= (ys % 1))
  {
  if  ((Math.ceil(xs) * Math.floor(ys) ) >= csize)
   {xpw = (int) Math.ceil(xs);ypw = (int) Math.floor(ys);}
  else
   {xpw = (int) Math.ceil(xs);ypw = (int) Math.ceil(ys); }
  }
  else
  {
   if  ((Math.floor(xs) * Math.ceil(ys) ) >= csize)
    {xpw = (int) Math.floor(xs);ypw = (int) Math.ceil(ys);}
   else
    {xpw = (int) Math.ceil(xs);ypw = (int) Math.ceil(ys); }
   }
  frmregion.clear();

  xrmd = wxcor%xpw;
  yrmd = wycor%ypw;
  for (int j=0 ; j<= ypw-1; ++j  )
  {
   for (int i=0 ; i<= xpw-1; ++i  )
   {
       x1 =   (wxcor / xpw)*i+1+ xrmd /2 ;
       y1 = (wycor / ypw)*j+1 + yrmd/2 ;
       x2 =   (wxcor / xpw)*(i+1)-1 + xrmd/2  ;
       y2 = (wycor / ypw)*(j+1)-1 + yrmd/2  ;

       frmregion.add(new region(x1,y1,x2,y2));
   }

  }
   Vector clus ;
   String s;
   mcolors clcol = new mcolors();

    maingraph.nodeAbt.clear();
    for(int i=0; i<= cluster.size()-1; i++){
     clus=(Vector) cluster.elementAt(i);
   // if(clus.size() >= 3){
       // Color ss =  ;
      for(int j=0; j<= clus.size()-1; j++){
          s = (String)clus.elementAt(j);
           //frmregion.get(i).xul;
         int p1=ThreadLocalRandom.current().nextInt(frmregion.get(i).xul + 2 , frmregion.get(i).xlr-2 );;
         int p2 =ThreadLocalRandom.current().nextInt(frmregion.get(i).yul + 2 , frmregion.get(i).ylr-2 );;

         graphColCor ccat = new graphColCor(p1, p2, mcolors.getcolor(i));
           if (maingraph.nodeAbt.get(s) == null)
             maingraph.nodeAbt.putIfAbsent(s, ccat);
          else
            maingraph.nodeAbt.get(s).setCol(Color.RED);
          }
         //}

      }


}

public   void setclustercoorcir(int wxcor , int wycor)
{
  int csize=0 ,x1=0,y1=0,x2=0,y2=0;
  double xs=0 , ys =0;
  int xpw=0, ypw = 0;
  int xrmd=0 , yrmd = 0;
 // csize = cluster.size();
  //csize = 35;
  Vector cs;
   for(int i=0; i<= cluster.size()-1; i++){
         cs=(Vector)cluster.elementAt(i);
        // if(cs.size() >= 3){ csize = csize+1;  }
   csize = csize+1;
   }

  xs = Math.sqrt(4*csize/3);
  ys = Math.sqrt(3*csize/4);
  //xs = Math.sqrt(6*csize/3);
  //ys = Math.sqrt(3*csize/5);

  if ((xs % 1) >= (ys % 1))
  {
  if  ((Math.ceil(xs) * Math.floor(ys) ) >= csize)
   {xpw = (int) Math.ceil(xs);ypw = (int) Math.floor(ys);}
  else
   {xpw = (int) Math.ceil(xs);ypw = (int) Math.ceil(ys); }
  }
  else
  {
   if  ((Math.floor(xs) * Math.ceil(ys) ) >= csize)
    {xpw = (int) Math.floor(xs);ypw = (int) Math.ceil(ys);}
   else
    {xpw = (int) Math.ceil(xs);ypw = (int) Math.ceil(ys); }
  }
 frmregion.clear();

 xrmd = wxcor%xpw;
 yrmd = wycor%ypw;
 for (int j=0 ; j<= ypw-1; ++j  )
 {
  for (int i=0 ; i<= xpw-1; ++i  )
  {
      x1 =   (wxcor / xpw)*i+1+ xrmd /2 ;
      y1 = (wycor / ypw)*j+1 + yrmd/2 ;
      x2 =   (wxcor / xpw)*(i+1)-1 + xrmd/2  ;
      y2 = (wycor / ypw)*(j+1)-1 + yrmd/2  ;

      frmregion.add(new region(x1,y1,x2,y2));
  }

 }
  float radius = (x2-x1)/2-2;
  float a = (x2-x1)/2-2;
  float b = (y2-y1)/2-2;

  float angle=0;
  Vector clus ;
  String s;
  mcolors clcol = new mcolors();

   maingraph.nodeAbt.clear();
   for(int i=0; i<= cluster.size()-1; i++){
    clus=(Vector) cluster.elementAt(i);
     for(int j=0; j<= clus.size()-1; j++){
        angle = (float) ((2*Math.PI/clus.size())*(j+1));
        s = (String)clus.elementAt(j);

       //int p1=ThreadLocalRandom.current().nextInt(frmregion.get(i).xul + 2 , frmregion.get(i).xlr-2 );;
       //int p2 =ThreadLocalRandom.current().nextInt(frmregion.get(i).yul + 2 , frmregion.get(i).ylr-2 );;
      //   int r1 = ThreadLocalRandom.current().nextInt(Math.round(radius/3), Math.round(radius) );
        int p1  = (int) (frmregion.get(i).xul+(frmregion.get(i).xlr -frmregion.get(i).xul)/2 + a*Math.cos(angle));
        int p2  = (int) (frmregion.get(i).yul+(frmregion.get(i).ylr-frmregion.get(i).yul)/2 + b*Math.sin(angle));

       graphColCor ccat = new graphColCor(p1, p2, mcolors.getcolor(i));
         if (maingraph.nodeAbt.get(s) == null)
           maingraph.nodeAbt.putIfAbsent(s, ccat);
        else
          maingraph.nodeAbt.get(s).setCol(Color.RED);
        }
       //}

     }


}

public   void setclustercoorcirrand(int wxcor , int wycor)
{
 int csize=0 ,x1=0,y1=0,x2=0,y2=0;
 double xs=0 , ys =0;
 int xpw=0, ypw = 0;
 int xrmd=0 , yrmd = 0;
// csize = cluster.size();
 //csize = 35;
 Vector cs;
  for(int i=0; i<= cluster.size()-1; i++){
      cs=(Vector)cluster.elementAt(i);
      // if(cs.size() >= 3){ csize = csize+1;  }
  csize = csize+1;
  }

 xs = Math.sqrt(4*csize/3);
 ys = Math.sqrt(3*csize/4);
 //xs = Math.sqrt(6*csize/3);
 //ys = Math.sqrt(3*csize/5);

 if ((xs % 1) >= (ys % 1))
 {
 if  ((Math.ceil(xs) * Math.floor(ys) ) >= csize)
  {xpw = (int) Math.ceil(xs);ypw = (int) Math.floor(ys);}
 else
  {xpw = (int) Math.ceil(xs);ypw = (int) Math.ceil(ys); }
 }
 else
 {
  if  ((Math.floor(xs) * Math.ceil(ys) ) >= csize)
   {xpw = (int) Math.floor(xs);ypw = (int) Math.ceil(ys);}
  else
   {xpw = (int) Math.ceil(xs);ypw = (int) Math.ceil(ys); }
  }
 frmregion.clear();

 xrmd = wxcor%xpw;
 yrmd = wycor%ypw;
 for (int j=0 ; j<= ypw-1; ++j  )
 {
  for (int i=0 ; i<= xpw-1; ++i  )
  {
      x1 =   (wxcor / xpw)*i+1+ xrmd /2 ;
      y1 = (wycor / ypw)*j+1 + yrmd/2 ;
      x2 =   (wxcor / xpw)*(i+1)-1 + xrmd/2  ;
      y2 = (wycor / ypw)*(j+1)-1 + yrmd/2  ;

      frmregion.add(new region(x1,y1,x2,y2));
  }

 }
  float radius = (x2-x1)/2-2;
  float a = (x2-x1)/2-2;
  float b = (y2-y1)/2-2;

  float angle=0;
  Vector clus ;
  String s;
  mcolors clcol = new mcolors();

   maingraph.nodeAbt.clear();
   for(int i=0; i<= cluster.size()-1; i++){
    clus=(Vector) cluster.elementAt(i);
     for(int j=0; j<= clus.size()-1; j++){
        angle = (float) ((2*Math.PI/clus.size())*(j+1));
        s = (String)clus.elementAt(j);

       //int p1=ThreadLocalRandom.current().nextInt(frmregion.get(i).xul + 2 , frmregion.get(i).xlr-2 );;
       //int p2 =ThreadLocalRandom.current().nextInt(frmregion.get(i).yul + 2 , frmregion.get(i).ylr-2 );;
        int ra = ThreadLocalRandom.current().nextInt(Math.round(2*a/3), Math.round(a) );
        int rb = ThreadLocalRandom.current().nextInt(Math.round(2*b/3), Math.round(b) );

        int p1  = (int) (frmregion.get(i).xul+(frmregion.get(i).xlr -frmregion.get(i).xul)/2 + ra*Math.cos(angle));
        int p2  = (int) (frmregion.get(i).yul+(frmregion.get(i).ylr-frmregion.get(i).yul)/2 + rb*Math.sin(angle));
```

```
          graphColCor ccat = new graphColCor(p1, p2, mcolors.getcolor(i));
            if (maingraph.nodeAbt.get(s) == null)
              maingraph.nodeAbt.putIfAbsent(s, ccat);
           else
             maingraph.nodeAbt.get(s).setCol(Color.RED);
           }
          //}

        }


}



public  graph CopyGraph (graph G){

Vector NodeL = new Vector();
Hashtable AdjL = new Hashtable();
String nodeName;
Vector temp = new Vector();



for(int i=0; i<= G.NodeList.size()-1; i++){
nodeName = (String)G.NodeList.elementAt(i);
NodeL.add(nodeName);
temp = (Vector)(G.AdjList.get(nodeName));
Vector temp1 = new Vector();
for(int j=0; j<= temp.size()-1; j++)
temp1.add(temp.elementAt(j));

AdjL.put(nodeName, temp1);

}


graph copy = new graph(NodeL, AdjL);


return copy;
}

public   Vector createcluster (  double density, double cp ){
    AllClusters.clear();
    inputGraph= CopyGraph(maingraph);

    originalGraph= CopyGraph(maingraph);

    WeightClusters(density, cp);

     if(density < 0.7)
      originalGraph = CopyGraph(inputGraph);

     DegreeClusters(density, cp);
    // AllClusters.sort(new MyCompare());
    cluster = AllClusters;
     setclusterno();

return AllClusters;

}
```

```
public  void WeightClusters (double density, double cp){

int wMoreThanZero = 1;
double weight = 0;
int id=0;
String nodeName;
//graph inputGraph = CopyGraph(maingraph); ;

Vector Weights =  new Vector();
Vector nonZeroWNodes = new Vector();
int nl= inputGraph.NodeList.size();
for(int i=0; i<= nl-1; i++){
weight=inputGraph.NodeWeight((String)(inputGraph.NodeList.elementAt(i)));
if(weight > 0){
nonZeroWNodes.add(inputGraph.NodeList.elementAt(i));
Weights.add(new Double(weight));
}
}



int chk=0;
while ( wMoreThanZero == 1){

double max=0;
Vector seed = new Vector();
Vector FCluster = new Vector();
Vector Cluster = new Vector();
Vector AttachCluster = new Vector();
Vector neighbor = new Vector();

for(int i=0; i<= Weights.size()-1; i++){
weight= (Double)Weights.elementAt(i);
//System.out.println(weight);
if (weight > max){
max = weight;
id=i;
}
}
//System.out.println(max +"\t"+ id);

if(max==0)
wMoreThanZero = 0;

else{
nodeName = (String)nonZeroWNodes.elementAt(id);
//System.out.println(nodeName);
seed.add(nodeName);
//System.out.println((String)seed.elementAt(0));
FCluster = inputGraph.MakeCluster( seed, density, cp);
//System.out.println(FCluster.size());
Cluster = originalGraph.MakeCluster( FCluster, density, cp);

AllClusters.add(Cluster);
//System.out.println(inputGraph.EdgeCount());

inputGraph.RemoveClusterEdges(Cluster);

//System.out.println(inputGraph.EdgeCount());
//System.out.println(originalGraph.EdgeCount());
for(int i=0; i<= Cluster.size()-1; i++){
nodeName = (String)Cluster.elementAt(i);
if(nonZeroWNodes.contains(nodeName)){
//weight =inputGraph.NodeWeight(nodeName);
id = nonZeroWNodes.indexOf(nodeName);
//System.out.println(nodeName+"\t"+ id+"\t"+ weight);
//Weights.set(id, new Integer(weight));
nonZeroWNodes.removeElementAt(id);
Weights.removeElementAt(id);
}
}

neighbor = inputGraph.Neighbors(Cluster);

for(int i=0; i<= neighbor.size()-1; i++){
nodeName =(String)neighbor.elementAt(i);
if(nonZeroWNodes.contains(nodeName)){
weight =inputGraph.NodeWeight(nodeName);
id = nonZeroWNodes.indexOf(nodeName);
//System.out.println(nodeName+"\t"+ id+"\t"+ weight);
Weights.set(id, new Double(weight));
}
}

}

}

}


public   void DegreeClusters (  double density, double cp ){


int dMoreThanZero = 1;
int degree = 0;
int id=0;
String nodeName;
//Hashtable degreeTable = new Hashtable();
int nl= inputGraph.NodeList.size();
Vector Degrees = new Vector();
Vector AttachCluster = new Vector();
Vector nonZeroDNodes = new Vector();
Vector clus = new Vector();
String S;



for(int i=0; i<= nl-1; i++){
degree = inputGraph.NodeDegree((String)(inputGraph.NodeList.elementAt(i)));
if(degree > 0){
nonZeroDNodes.add(inputGraph.NodeList.elementAt(i));
Degrees.add(new Integer(degree));
//degreeTable.put((String)(inputGraph.NodeList.elementAt(i)), new Integer(degree));
}
}

int chk=0;
while ( dMoreThanZero == 1){

int max=0;
Vector seed = new Vector();
Vector FCluster = new Vector();
Vector Cluster = new Vector();


for(int i=0; i<= Degrees.size()-1; i++){
degree= (Integer)Degrees.elementAt(i);
```

```
if (degree > max){
max = degree;
id=i;
}
}

if(max==0)
dMoreThanZero = 0;

else{
nodeName= (String)nonZeroDNodes.elementAt(id);
seed.add(nodeName);
FCluster = inputGraph.MakeCluster(seed, density, cp);
Cluster = originalGraph.MakeCluster(FCluster, density, cp);
AllClusters.add(Cluster);
inputGraph.RemoveClusterEdges(Cluster);

for(int i=0; i<= Cluster.size()-1; i++){
nodeName = (String)Cluster.elementAt(i);
if(nonZeroDNodes.contains(nodeName)){
id = nonZeroDNodes.indexOf(nodeName);
nonZeroDNodes.removeElementAt(id);
Degrees.removeElementAt(id);
}

}

}


}
  cluster = AllClusters;
  cluster.sort(new MyCompare());
 //return AllClusters;
}




 public   graph GraphRead (String FN) throws IOException{

    Vector edgeL = new Vector();
    String filename = FN;
    int intr = 0;
    BufferedReader stdin = new BufferedReader(new FileReader(filename));
    String Interaction;
    while((Interaction = stdin.readLine()) != null){
      edgeL.add(Interaction);
      intr= intr+1;
    }
    stdin.close();
    String[][] Protintr1 = new String[intr][2];
    String[][] Protintr2 = new String[intr][2];
    int kk = 0;
    int mm=0;
    StringTokenizer tok;
    for(int i=0; i<=intr-1; i++){
      tok = new StringTokenizer((String)edgeL.get(i), "\t");
      Protintr1[i][0] = tok.nextToken().trim();
      Protintr1[i][1] = tok.nextToken().trim();
    }
    for(int i=0; i<=intr-1; i++){
      mm =0;
      for(int j=0; j<=kk; j++){
        //System.out.println(m);
        if ((Protintr1[i][0].equals(Protintr1[i][1])) | | (Protintr1[i][0].equals(Protintr2[j][0])) &&
(Protintr1[i][1].equals(Protintr2[j][1])) | | (Protintr1[i][0].equals(Protintr2[j][1])) &&
(Protintr1[i][1].equals(Protintr2[j][0]))){
              mm=1;
          break;
        }
      }
      if(mm == 0){
        for(int ss=0; ss<=1; ss++){
          Protintr2[kk][ss]= Protintr1[i][ss];
          //System.out.print(Protintr2[kk][ss]+"\t");
        }
        kk=kk+1;
      }
    }

    intr = kk;
Vector NodeList = new Vector();
String A;
String B;


for(int i=0; i<=intr-1; i++){
A= Protintr2[i][0];
B= Protintr2[i][1];

if (!(NodeList.contains(A)))
NodeList.add(A);
if (!(NodeList.contains(B)))
NodeList.add(B);
}

Hashtable AdjList = new Hashtable();
for(int i=0; i<=NodeList.size()-1; i++){
 Vector neighborList = new Vector();
  A=(String)(NodeList.elementAt(i));
  for(int j=0; j<=intr-1; j++){
   if ((A.equals(Protintr2[j][0])))
     neighborList.add(Protintr2[j][1]);
   if ((A.equals(Protintr2[j][1])))
    neighborList.add(Protintr2[j][0]);
    }
  AdjList.put(A, neighborList);
 }
 maingraph = new graph(NodeList, AdjList);

 return maingraph;
}


public   graph GraphReadal (ArrayList fn ) {

    Vector edgeL = new Vector();
  //  String filename = FN;
    int intr = 0;
   // BufferedReader stdin = new BufferedReader(new FileReader(filename));
    String Interaction;
    for (Object temp : fn) {
             edgeL.add(temp.toString());
        intr= intr+1;

          }
    /*
    while((Interaction = stdin.readLine()) != null){
      edgeL.add(Interaction);
      intr= intr+1;
    }
    */
//stdin.close();
    String[][] Protintr1 = new String[intr][2];
    String[][] Protintr2 = new String[intr][2];
    int kk = 0;
    int mm=0;
    StringTokenizer tok;
    for(int i=0; i<=intr-1; i++){
      tok = new StringTokenizer((String)edgeL.get(i), "\t");
      Protintr1[i][0] = tok.nextToken().trim();
      Protintr1[i][1] = tok.nextToken().trim();
    }
    for(int i=0; i<=intr-1; i++){
      mm =0;
      for(int j=0; j<=kk; j++){
        //System.out.println(m);
        if ((Protintr1[i][0].equals(Protintr1[i][1])) || (Protintr1[i][0].equals(Protintr2[j][0])) &&
(Protintr1[i][1].equals(Protintr2[j][1])) || (Protintr1[i][0].equals(Protintr2[j][1])) &&
(Protintr1[i][1].equals(Protintr2[j][0]))){
              mm=1;
          break;
        }
      }
      if(mm == 0){
        for(int ss=0; ss<=1; ss++){
          Protintr2[kk][ss]= Protintr1[i][ss];
          //System.out.print(Protintr2[kk][ss]+"\t");
        }
        kk=kk+1;
      }
    }

    intr = kk;
Vector NodeList = new Vector();
String A;
String B;


for(int i=0; i<=intr-1; i++){
A= Protintr2[i][0];
B= Protintr2[i][1];

if (!(NodeList.contains(A)))
NodeList.add(A);
if (!(NodeList.contains(B)))
NodeList.add(B);
}

Hashtable AdjList = new Hashtable();
for(int i=0; i<=NodeList.size()-1; i++){
 Vector neighborList = new Vector();
  A=(String)(NodeList.elementAt(i));
  for(int j=0; j<=intr-1; j++){
   if ((A.equals(Protintr2[j][0])))
     neighborList.add(Protintr2[j][1]);
   if ((A.equals(Protintr2[j][1])))
    neighborList.add(Protintr2[j][0]);
    }
  AdjList.put(A, neighborList);
 }
 maingraph = new graph(NodeList, AdjList);

 return maingraph;
}

public  void graphset(graph gph)
{
   maingraph = new graph(gph.NodeList, gph.AdjList);
   maingraph.setcoordinate(clsWindow.xwidth-10, clsWindow.yheight-25);

}


}
```

This class contains a graph with nodes, adjacent list of each node, color and coordinate of each node.

```
package com.bb.graph;

import java.awt.Color;
import java.io.*;
import java.util.*;
import java.util.concurrent.ThreadLocalRandom;
import java.util.zip.Adler32;
import javax.swing.JOptionPane;

class graph implements  Serializable{

Vector NodeList;
Hashtable AdjList;
Hashtable<String , graphColCor> nodeAbt;

public graph()
{
 NodeList =new Vector();
 AdjList = new Hashtable();
 nodeAbt = new Hashtable<String , graphColCor>();
}
public graph( graph gh )
{
  this.NodeList = gh.NodeList;
  this.AdjList = gh.AdjList;
  this.nodeAbt = gh.nodeAbt;
}
public graph(Vector NL, Hashtable AdjL){

NodeList= NL;
AdjList=AdjL;
```

```
}

public  graph(Vector NL, Hashtable AdjL , Hashtable nbt ){
 NodeList =new Vector();
 AdjList = new Hashtable();
 nodeAbt = new Hashtable<String , graphColCor>();
NodeList= (Vector) NL.clone();
AdjList=(Hashtable) AdjL.clone();
nodeAbt = (Hashtable<String, graphColCor>) nbt.clone();
}

public void setdata(Vector NL, Hashtable AdjL , Hashtable nbt  )
{
 this.NodeList =new Vector();
 this.AdjList = new Hashtable();
 this.nodeAbt = new Hashtable<String , graphColCor>();
 Vector ad ;
 graphColCor gcc = new graphColCor();
   for (int i=0; i<=NL.size()-1 ; ++i)
   {
     this.NodeList.add(NL.get(i));
     ad=(Vector) AdjL.get(NL.get(i));
     this.AdjList.put(NL.get(i), ad);
     gcc = (graphColCor)  nbt.get(NL.get(i));
     this.nodeAbt.put((String) NL.get(i), gcc);
   }

}

public void setdataall(myDCluster dc , int rpos)
    {
       Vector cnode = new Vector();
       Vector ad = new Vector();
       graphColCor gcc = new graphColCor();

       if (rpos >=0)
       {
        cnode=(Vector) dc.clusterd.elementAt(rpos);
        for (int i=0; i<= cnode.size()-1; ++i)
         {
            gcc = new graphColCor();
            ad = new Vector();

            ad=(Vector) dc.maingraphd.AdjList.get(cnode.get(i));
            this.AdjList.put(cnode.get(i), ad);
            gcc = (graphColCor)  dc.maingraphd.nodeAbt.get(cnode.get(i));
            this.nodeAbt.put((String) cnode.get(i), gcc);
            this.NodeList.add(cnode.get(i));
         }

       }
    }


public String selectnode(int x , int y)
{
  String sk="";
  for (String key : this.nodeAbt.keySet()) {
       graphColCor  temp = (graphColCor)  nodeAbt.get(key);
         if (((x > temp.xpos-5) && (x < temp.xpos+5)) && ((y > temp.ypos-5) && (y < temp.ypos+5)))
            sk = key;
  }
  return sk;
}

public void changenodecoorsel(String sk, int dx , int dy)
{
     if   (this.nodeAbt.get(sk) != null)
     {
      nodeAbt.get(sk).xpos= nodeAbt.get(sk).xpos+dx;
      nodeAbt.get(sk).ypos= nodeAbt.get(sk).ypos+dy;
     }

}


public void changenodecoor(int x , int y , int dx , int dy)
{
    graphColCor temp , to;
    Set<String> keys = this.nodeAbt.keySet();
    for(String key: keys){
     temp = (graphColCor)  nodeAbt.get(key);
     if (((x > temp.xpos-5) && (x < temp.xpos+5)) && ((y > temp.ypos-5) && (y < temp.ypos+5)))
     {
      this.nodeAbt.get(key).setXpos(temp.xpos+dx);
      this.nodeAbt.get(key).setYpos(temp.ypos+dy);
      break;
     }
    }
}

public void megnifynodecoor( region r )
{
    graphColCor temp = new graphColCor();
    Set<String> keys = this.nodeAbt.keySet();
    for(String key: keys){
     temp = (graphColCor)  this.nodeAbt.get(key);
      this.nodeAbt.get(key).setXpos((temp.xpos - r.xul)*600/(r.xlr-r.xul));
      this.nodeAbt.get(key).setYpos((temp.ypos - r.yul)*600/(r.ylr-r.yul));

    }

}

public void megnifynodecoorreshape( region r )
{
    graphColCor temp = new graphColCor();
    Set<String> keys = this.nodeAbt.keySet();
    for(String key: keys){
      temp = (graphColCor)  this.nodeAbt.get(key);
      if (temp.xpos < 0)
        this.nodeAbt.get(key).setXpos(r.xlr -((r.xul-temp.xpos)%(r.xlr-r.xul)) );
      if (temp.xpos > 600)
        this.nodeAbt.get(key).setXpos(r.xul + ((temp.xpos-r.xlr)%(r.xlr-r.xul)));
      if (temp.ypos < 0)
        this.nodeAbt.get(key).setYpos(r.ylr -((r.yul-temp.ypos)%(r.ylr-r.yul)) );
      if (temp.ypos > 600)
        this.nodeAbt.get(key).setYpos(r.yul + ((temp.ypos-r.ylr)%(r.ylr-r.yul)));
    }
}


public void setcoordinatesub(){

    graphColCor temp = new graphColCor();
    Set<String> keys = this.nodeAbt.keySet();
```

```
int i=0,j=0,k=0,k1=0, p=0,loop=1;
 float radius = 150;
  float a = 100;
  float b = 100;
  float angle =0 ;
  float angle1 = 0;
for(String key: keys){

    temp = (graphColCor)  this.nodeAbt.get(key);

   if (clsWindow.grpview == 1) {
     i =ThreadLocalRandom.current().nextInt(50, 550 );
     j = ThreadLocalRandom.current().nextInt(50, 550);
     }
   else if (clsWindow.grpview == 2) {
      angle = (float) ((2*Math.PI/this.nodeAbt.size())*(k+1));
      ++k;
       i = (int )( 300+ (a+150)*Math.cos(angle));
       j = (int )( 300+ (b+150)*Math.sin(angle));


   }

   else if (clsWindow.grpview == 3) {

       /*
     if ((loop%3)==0)
      {
         angle = (float) ((2*Math.PI/(this.nodeAbt.size()/3))*(k+1));
         i = (int )( 300+ a*Math.cos(angle));
         j = (int )( 300+ b*Math.sin(angle));

         ++k;
      }
      else
      {
        angle1 = (float) ((2*Math.PI/(this.nodeAbt.size()-this.nodeAbt.size()/3))*(k1+1));
          i = (int )( 300+ (a+150)*Math.cos(angle1));
          j = (int )( 300+ (b+150)*Math.sin(angle1));

        ++k1;

      }
       ++loop;

    */
     angle = (float) ((2*Math.PI/this.nodeAbt.size())*(k+1));
      ++k;

      p =ThreadLocalRandom.current().nextInt(0, 2);

     if (p==1)
      {
       i = (int )( 300+ (a+50)*Math.cos(angle));
       j = (int )( 300+ (b+50)*Math.sin(angle));
      }
     else
      {
       i = (int )( 300+ (a+150)*Math.cos(angle));
       j = (int )( 300+ (b+150)*Math.sin(angle));

      }

   }
```

```
       else if (clsWindow.grpview == 4) {
          angle = (float) ((2*Math.PI/this.nodeAbt.size())*(k+1));
          ++k;
        if ((k%3)==0)
        {
         i = (int )( 300+ a*Math.cos(angle));
         j = (int )( 300+ b*Math.sin(angle));
        }
        else if ((k%3)==1)
        {
         i = (int )( 300+ (a+75)*Math.cos(angle));
         j = (int )( 300+ (b+75)*Math.sin(angle));
        }
        else if ((k%3)==2)
        {
         i = (int )( 300+ (a+150)*Math.cos(angle));
         j = (int )( 300+ (b+150)*Math.sin(angle));
        }

       }


       this.nodeAbt.get(key).setXpos(i);
        this.nodeAbt.get(key).setYpos(j);

    }
  }



public void setcoordinate(int xwidth , int ywidth){
   Enumeration vEnum = this.NodeList.elements();
    Random randomGenerator = new Random();
    nodeAbt = new Hashtable();
     while(vEnum.hasMoreElements())
      {

       String n = (String) vEnum.nextElement();
       //int i =ThreadLocalRandom.current().nextInt(50, 750 + 1);
       //int j = ThreadLocalRandom.current().nextInt(50, 450 + 1);
       int i =ThreadLocalRandom.current().nextInt(50, xwidth-50);
       int j = ThreadLocalRandom.current().nextInt(50, ywidth-50);

       graphColCor ccat = new graphColCor(i, j, Color.BLUE );

       nodeAbt.put(n, ccat);


    }

}

public void RemoveEdge(String A, String B){

Vector temp = new Vector();

temp = (Vector) AdjList.get(A);

temp.remove("B");

temp = (Vector) AdjList.get(B);

temp.remove("A");
```

```
}


public void RemoveNode(String A){

NodeList.remove(A);
AdjList.remove(A);

Vector temp = new Vector();
Enumeration e = AdjList.elements();


while( e. hasMoreElements() ){
temp = (Vector)e.nextElement();
temp.remove(A);
}

}


public Vector Neighbors(Vector A){

int l=0;
int nl=0;

l= A.size();


Vector temp = new Vector();
Vector neigh = new Vector();
for(int i=0; i<= l-1; i++){
temp=(Vector)AdjList.get((String)(A.elementAt(i)));
nl= temp.size();
//System.out.println(nl);
for(int j=0; j<= nl-1; j++){
if(!(neigh.contains(temp.elementAt(j))))
neigh.add(temp.elementAt(j));
}
}


for(int i=0; i<= l-1; i++)
neigh.remove(A.elementAt(i));

return neigh;


}



public double EdgeWeight(String A, String B ){

int weight=0;
double eWeight = 0;
Vector temp1 = new Vector();
Vector temp2 = new Vector();

temp1=(Vector)AdjList.get(A);
temp2=(Vector)AdjList.get(B);

for(int i=0; i<= temp1.size()-1; i++){
if(temp2.contains(temp1.elementAt(i)))
weight=weight+1;
}

eWeight = (double)weight/(temp1.size()*temp2.size());

return eWeight;

}



public double NodeBond(String A, String B ){

int weight=0;
double eWeight = 0;
Vector temp1 = new Vector();
Vector temp2 = new Vector();

temp1=(Vector)AdjList.get(A);
temp2=(Vector)AdjList.get(B);

if(temp2.contains(A)){
for(int i=0; i<= temp1.size()-1; i++){
if(temp2.contains(temp1.elementAt(i)))
weight=weight+1;
}

eWeight = (double)weight/(temp1.size()*temp2.size());
}

return eWeight;

}


public double NodeWeight(String A){

Vector temp = new Vector();
String B;
double weight=0;

temp=(Vector)AdjList.get(A);

for(int i=0; i<= temp.size()-1; i++){
B=(String)temp.elementAt(i);
weight=weight+EdgeWeight(A, B);
}


return weight;

}



public int NodeDegree(String A){

Vector temp = new Vector();
String B;
int degree=0;

temp=(Vector)AdjList.get(A);

degree = temp.size();
```

```
return degree;

}



public Vector MakeClusterold( Vector A, double din, double cpin){

Vector cluster= new Vector();
Vector neighbor= new Vector();
Vector mWeightNeigh = new Vector();
Vector hiLiNei= new Vector();
Vector hiLiWei= new Vector();



double wsum=0;
double maxweight=0;
int connect=0;
int maxlink=0;
int id=0;
String priority="pr";
String s;
double d=0.0;
double dd=0.0;
double cpp=0;
int cl=0;
int whileloop=0;
double maxl=0;
int c =0;



for(int i=0; i<= A.size()-1; i++)
cluster.add(A.elementAt(i));

//System.out.println(A.size() +"\t"+ A.elementAt(0));//####################
int tt=0;
//System.out.println(cluster.size());

while (whileloop == 0){

//tt=tt+1;
//if(tt==4)
//whileloop=1;

//System.out.println(tt);
neighbor = Neighbors(cluster);
if(neighbor.size() == 0){
whileloop =1;
}
else{
//System.out.println(neighbor.size());
double weight[] = new double[neighbor.size()];

for(int i=0; i<= neighbor.size()-1; i++){
for(int j=0; j<= cluster.size()-1; j++){
wsum = wsum + NodeBond((String)(neighbor.elementAt(i)) , (String)(cluster.elementAt(j)));
}
weight[i]= wsum;
wsum=0;
}
```

```
maxweight = max(weight);

//System.out.println(neighbor.size()+"\t"+maxweight);//####################

//whileloop = 1;

//System.out.println(priority);//####################

for(int i=0; i<= neighbor.size()-1; i++){
if(weight[i] == maxweight){
priority = (String)(neighbor.get(i));
break;
}
}
cl= cluster.size();
d = density(cluster);
connect = links(cluster, priority);
cpp=connect/(d*cl);
//System.out.println("ClusterSize before adding priority
"+cluster.size()+"\t"+dd+"\t"+cpp);//####################
//System.out.println(cluster.elementAt(0));
cluster.addElement(priority);

dd= density(cluster);
//System.out.println("ClusterSize after adding priority
"+cluster.size()+"\t"+dd+"\t"+cpp);//####################
//System.out.println(cluster.elementAt(1));
//whileloop =1;
        if(dd < din || cpp < cpin){
cluster.remove(priority);
hiLiNei.clear();
hiLiWei.clear();
for(int i=0; i<= neighbor.size()-1; i++){
c = links(cluster, (String)neighbor.elementAt(i));
if(c > connect){
hiLiNei.add(neighbor.elementAt(i));
hiLiWei.add(new Double(weight[i]));
}
}

if(hiLiNei.size() == 0)
whileloop =1;
                                    else{
id=0;
maxl=0;
for(int i=0; i<= hiLiWei.size()-1; i++){
if (maxl < (Double)hiLiWei.elementAt(i)){
maxl = (Double)(hiLiWei.elementAt(i));
id = i;
}
}
priority = (String)hiLiNei.elementAt(id);
cl= cluster.size();
d = density(cluster);
connect = links(cluster, priority);
cpp=connect/(d*cl);
//System.out.println("ClusterSize before adding priority
"+cluster.size()+"\t"+dd+"\t"+cpp);//####################
//System.out.println(cluster.elementAt(0));
cluster.addElement(priority);

dd= density(cluster);
```

```
//System.out.println("ClusterSize after adding priority
"+cluster.size()+"\t"+dd+"\t"+cpp);//####################
//System.out.println(cluster.elementAt(1));
//whileloop =1;
if(dd < din || cpp < cpin){
cluster.remove(priority);
whileloop =1;
}
                                                  }

}
//cluster.add("D");
//System.out.println("ClusterSize after adding priority  "+cluster.size()+"\t"+dd+"\t"+cpp);
}
}
return cluster;

}



public Vector MakeCluster( Vector A, double din, double cpin){

//System.out.println("MakeCluster has been called");//####################

Vector cluster= new Vector();
Vector neighbor= new Vector();
Vector mWeightNeigh = new Vector();
Vector hiLiNei= new Vector();
Vector hiLiWei= new Vector();



double wsum=0;
double maxweight=0;
int connect=0;
int maxlink=0;
int id=0;
String priority="pr";
String s;
double d=0.0;
double dd=0.0;
double cpp=0;
int cl=0;
int whileloop=0;
double maxl=0;
int c =0;



for(int i=0; i<= A.size()-1; i++)
cluster.add(A.elementAt(i));

int tt=0;

while (whileloop == 0){
  neighbor = Neighbors(cluster);
   if(neighbor.size() == 0){
   whileloop =1;
   }
 else{
 double weight[] = new double[neighbor.size()];
 for(int i=0; i<= neighbor.size()-1; i++){
     for(int j=0; j<= cluster.size()-1; j++){
```

```
     wsum = wsum + NodeBond((String)(neighbor.elementAt(i)) , (String)(cluster.elementAt(j)));
     }
    weight[i]= wsum;
    wsum=0;
    }
maxweight = max(weight);
for(int i=0; i<= neighbor.size()-1; i++){
   if(weight[i] == maxweight){
     priority = (String)(neighbor.get(i));
     break;
  }
 }
cl= cluster.size();
d = density(cluster);
connect = links(cluster, priority);
cpp=connect/(d*cl);

cluster.addElement(priority);

dd= density(cluster);
       if(dd < din | | cpp < cpin){
cluster.remove(priority);

hiLiNei.clear();
hiLiWei.clear();

for(int i=0; i<= neighbor.size()-1; i++){
c = links(cluster, (String)neighbor.elementAt(i));
if(c > connect){
hiLiNei.add(neighbor.elementAt(i));
hiLiWei.add(new Double(weight[i]));
}
}

  if(hiLiNei.size() == 0)
     whileloop =1;
else{
     maxl=0;
     id =-1 ;
    for(int i=0; i<= hiLiWei.size()-1; i++) {
       if (maxl <= (Double)hiLiWei.elementAt(i)){
           maxl = (Double)(hiLiWei.elementAt(i));
           id = i;
          }
     }
  //  if (id == -1) break;

    priority = (String)hiLiNei.elementAt(id);

    cl= cluster.size();
     d = density(cluster);
     connect = links(cluster, priority);
     cpp=connect/(d*cl);


     cluster.addElement(priority);
     dd= density(cluster);
     if(dd < din | | cpp < cpin){
       cluster.remove(priority);
       whileloop =1;
      }
    }
 }
```

```
}

}
return cluster;

}


public double max(double[] Arry){

double k = Arry[0];
for(int i=0; i<= Arry.length-1; i++){
if (k < Arry[i])
k=Arry[i];
}

return k;

}





public int links(Vector A, String B){

int k=0;

Vector temp = new Vector();

for(int i=0; i<= A.size()-1; i++){
temp=(Vector)AdjList.get(A.elementAt(i));
if(temp.contains(B))
k=k+1;
}

return k;

}



public double density(Vector A){

int k= A.size();
Vector copyA = new Vector();
//System.out.println(k);
for(int i=0; i<= k-1; i++)
copyA.add(A.elementAt(i));

if(k==1){
return 1.0;
}

else{
int kk=0;
double d=0.0;
Vector temp = new Vector();
String s;


for(int i=0; i<= k-2; i++){
s=(String)(copyA.elementAt(0));
```

```
copyA.remove(s);
temp=(Vector)AdjList.get(s);
for(int j=0; j<= copyA.size()-1; j++){
if(temp.contains(copyA.elementAt(j)))
kk=kk+1;
}
}
//System.out.println(kk);
d=(2.0*kk)/(k*(k-1));

return d;
}

}


public void RemoveClusterEdges(Vector A){

int cl= A.size();
String n1;
String n2;
Vector temp = new Vector();


for(int i=0; i<= cl-1; i++){
n1 = (String)(A.elementAt(i));
temp = (Vector)AdjList.get(n1);
//System.out.println(n1 +temp.size());
for(int j=0; j<= cl-1; j++){
n2 = (String)(A.elementAt(j));
temp.remove(n2);
//System.out.println(n2+temp.size());
}
}

}


public void RemoveClusterNodes(Vector A){

int cl= A.size();
String nodeName;

for(int i=0; i<= cl-2; i++){
nodeName = (String)(A.elementAt(i));
RemoveNode(nodeName);
}

}


public int EdgeCount( ){

int NoOfEdges = 0;
Vector temp = new Vector();
String s;


for(int i=0; i<= NodeList.size()-1; i++){
s=(String)(NodeList.elementAt(i));
temp=(Vector)AdjList.get(s);
NoOfEdges = NoOfEdges+temp.size();
}
```

```
NoOfEdges = NoOfEdges/2;

return NoOfEdges;

}


public Vector AddAttach( Vector A, double affinity){

Vector B = new Vector();
double aff1=0;
double aff2=0;
int S = A.size();

for(int i=0; i<= A.size()-1; i++){
B.add(A.elementAt(i));
}

if (B.size() <= 2)
return B;
else{
int degree=0;
int connect=0;
Vector ClusNeighbors= new Vector();
String temp;

ClusNeighbors = Neighbors(A);

for(int i=0; i<= ClusNeighbors.size()-1; i++){

temp = (String)ClusNeighbors.elementAt(i);
connect = links(A, temp);
degree = NodeDegree(temp);

aff1 = (double)(connect/degree);
aff2 = (double)(connect/S);

if (aff2 > 0.3 && aff1 >= affinity)
B.add(temp);
}
return B;
}
}
}
/*
 * DPClusSBO1.1 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * DPClusSBO1.1 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */
```

This class is used to maintain the coordicnate and layout of different bicluster.

```
package com.bb.graph;

import java.awt.Color;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;
import java.util.SortedSet;
import java.util.TreeSet;
import java.util.Vector;

/**
 *
 * @author mohammad bozlulkarim
 */
public class myClusterbi implements Serializable{

   HashMap <String, ArrayList<String>> adlistf=new HashMap<>();;
   HashMap <String, ArrayList<String>> adlists=new HashMap<>();;
   HashMap <String, graphColCor> nodeAbt= new HashMap<>();
   HashMap<Integer, clsBinode> bclus=new HashMap<>();;
   ArrayList<region> frmregion = new ArrayList<region>();
   ArrayList<Integer> frmregionkey = new ArrayList<>();


   public myClusterbi() {

   }

 myClusterbi(myClusterbi mcb)
 {
     this.adlistf = new HashMap<>(mcb.adlistf) ;
     this.adlists = new HashMap<>(mcb.adlists);
     this.nodeAbt = new HashMap<>(mcb.nodeAbt);
     this.frmregion= new ArrayList<region>(mcb.frmregion);
     this.frmregionkey=new ArrayList<>(mcb.frmregionkey);
     this.bclus  = new HashMap<>(mcb.bclus);

}
 myClusterbi( clsMatrixData cmatx, clsBicluster bcls )
 {
   bclus = new HashMap<>();
   bclus = bcls.getBclus();
   adlistf = new HashMap<>();
   adlists = new HashMap<>();
   String s1,s2;

   Set xxs = new HashSet();
   Set yys = new HashSet();
   ArrayList xx = new ArrayList();
   ArrayList yy = new ArrayList();
   ArrayList adl = new ArrayList();

   Set ad = new HashSet();

   for (Integer key : bclus.keySet()) {
      xxs.addAll(bclus.get(key).getFtuple());
      yys.addAll(bclus.get(key).getStuple());
    }
   xx = new ArrayList(xxs);
   yy = new ArrayList(yys);

    for (int i = 0; i < xx.size(); i++) {
          s1 = (String) xx.get(i);
          ad = new HashSet();
          for (int j = 0; j < yy.size(); j++) {
              s2 = (String) yy.get(j);
               if (cmatx.getmatdata(s1, s2)==1) ad.add(s2);
             }
           adl = new ArrayList(ad);

           adlistf.put(s1, adl);
         }
       for (int i = 0; i < yy.size(); i++) {
            s1 = (String) yy.get(i);
            ad = new HashSet();
            for (int j = 0;  j< xx.size(); j++) {
             s2 = (String) xx.get(j);
              if (cmatx.getmatdata(s2, s1)==1) ad.add(s2);
             }
          adl = new ArrayList(ad);
          adlists.put(s1, adl);
         }
   }

public String selectnode(int x , int y)
{
  String sk="";
  for (String key : this.nodeAbt.keySet()) {
       graphColCor  temp = (graphColCor)  nodeAbt.get(key);
         if (((x > temp.xpos-5) && (x < temp.xpos+5)) && ((y > temp.ypos-5) && (y < temp.ypos+5)))
            sk = key;
  }

  return sk;
}

public void changenodecoorsel(String sk, int dx , int dy)
{
     if   (this.nodeAbt.get(sk) != null)
     {
      nodeAbt.get(sk).xpos= nodeAbt.get(sk).xpos+dx;
      nodeAbt.get(sk).ypos= nodeAbt.get(sk).ypos+dy;
     }

}

 public int findregion(int x , int y)
    {
     int rpos=-1;
     for(int i=0 ; i<=frmregion.size()-1; ++i)
    {
         region r = (region) frmregion.get(i);
     if (((x > r.xul+5) && (x < r.xlr-5)) && ((y > r.yul+5) && (y < r.ylr-5)))
         rpos = i;
     }

     return rpos;
 //    return frmregionkey.get(rpos+1) ;
    }

 public void megnifynodecoor( int rpos )
{
    Integer clsno = frmregionkey.get(rpos);
    ArrayList<String> clsnoes = new ArrayList<>();
    clsnoes.addAll(bclus.get(clsno).getFtuple());
    clsnoes.addAll(bclus.get(clsno).getStuple());
    graphColCor temp = new graphColCor();
    region r = frmregion.get(rpos);
    for (int i = 0; i < clsnoes.size(); i++) {
      String key = clsnoes.get(i);
      temp = (graphColCor)  this.nodeAbt.get(key);
      this.nodeAbt.get(key).setXpos((temp.xpos - r.xul)*750/(r.xlr-r.xul));
      this.nodeAbt.get(key).setYpos((temp.ypos - r.yul)*350/(r.ylr-r.yul));

    }

}

public void setcoordinatesub(int rpos)
{
    Integer clsno = frmregionkey.get(rpos);
    ArrayList<String> clsnoes = new ArrayList<>();
   int divs ;
   int p1,p2;
    clsnoes.addAll(bclus.get(clsno).getFtuple());

   divs = (int) 750/(clsnoes.size());
   p1 = 25 + divs/2;
   p2 = 100;
    for (int i = 0; i < clsnoes.size(); i++) {
      String key = clsnoes.get(i);
      this.nodeAbt.get(key).setXpos(p1);
      this.nodeAbt.get(key).setYpos(p2);
      p1 = p1+divs;
   }
   clsnoes.clear();
   clsnoes.addAll(bclus.get(clsno).getStuple());
  divs = (int) 750/(clsnoes.size());
   p1 = 25 + divs/2;
   p2 = 300;

    for (int i = 0; i < clsnoes.size(); i++) {
      String key = clsnoes.get(i);
      this.nodeAbt.get(key).setXpos(p1);
      this.nodeAbt.get(key).setYpos(p2);
      p1 = p1+divs;
   }

}

public void setcoordinate(int wxcor , int wycor)
 {
  int csize=0 ,x1=0,y1=0,x2=0,y2=0;
  double xs=0 , ys =0;
  int xpw=0, ypw = 0;
  int xrmd=0 , yrmd = 0;
  csize = bclus.size();
  xs = Math.sqrt(4*csize/3);
  ys = Math.sqrt(3*csize/4);

  if ((xs % 1) >= (ys % 1))
  {
  if  ((Math.ceil(xs) * Math.floor(ys) ) >= csize)
   {xpw = (int) Math.ceil(xs);ypw = (int) Math.floor(ys);}
  else
   {xpw = (int) Math.ceil(xs);ypw = (int) Math.ceil(ys); }
  }
  else
  {
   if  ((Math.floor(xs) * Math.ceil(ys) ) >= csize)
    {xpw = (int) Math.floor(xs);ypw = (int) Math.ceil(ys);}
   else
    {xpw = (int) Math.ceil(xs);ypw = (int) Math.ceil(ys); }
 }
xrmd = wxcor%xpw;
yrmd = wycor%ypw;

for (int j=0 ; j<= ypw-1; ++j  )
{
 for (int i=0 ; i<= xpw-1; ++i  )
 {
      x1 =   (wxcor / xpw)*i+1+ xrmd /2 ;
      y1 = (wycor / ypw)*j+1 + yrmd/2 ;
      x2 =   (wxcor / xpw)*(i+1)-1 + xrmd/2  ;
      y2 = (wycor / ypw)*(j+1)-1 + yrmd/2  ;
      frmregion.add(new region(x1,y1,x2,y2));
 }
}
float radius = (x2-x1)/2-2;
float a = (x2-x1)/2-2;
float b = (y2-y1)/8-2;
String   s1;
float angle=0;
ArrayList xx, yy;
int j=0;
 for(Integer key : bclus.keySet())
   {
     xx = new ArrayList(bclus.get(key).getFtuple());
    for (int i = 0; i < xx.size(); i++) {
      s1 = (String) xx.get(i);
      angle = (float) ((2*Math.PI/xx.size())*(i+1));
     int p1  = (int) (frmregion.get(j).xul+(frmregion.get(j).xlr -frmregion.get(j).xul)/2 + a*Math.cos(angle));
     int p2  = (int) (frmregion.get(j).yul+(frmregion.get(j).ylr-frmregion.get(j).yul)/8 + b*Math.sin(angle));
      graphColCor ccat = new graphColCor(p1, p2, Color.BLUE);
      if (nodeAbt.get(s1)!= null)
         nodeAbt.get(s1).Col =  Color.RED;
      else
        nodeAbt.put(s1, ccat);
       }
     yy = new ArrayList(bclus.get(key).getStuple());

     for (int i = 0; i < yy.size(); i++) {
      s1 = (String) yy.get(i);
      angle = (float) ((2*Math.PI/yy.size())*(i+1));
     int p1  = (int) (frmregion.get(j).xul+(frmregion.get(j).xlr -frmregion.get(j).xul)/2 + a*Math.cos(angle));
     int p2  = (int) (frmregion.get(j).yul+(frmregion.get(j).ylr-frmregion.get(j).yul)*7/8 + b*Math.sin(angle));
      graphColCor ccat = new graphColCor(p1, p2, Color.GREEN);
      if (nodeAbt.get(s1)!= null)
         nodeAbt.get(s1).Col =  Color.RED;
      else
        nodeAbt.put(s1, ccat);
       }

     frmregionkey.add(key);


    ++j;

     }
  // System.out.println("");

 }


}

/*
```

```
 * DPClusSBO1.1 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * DPClusSBO1.1 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */
```

This class is used for bipartite graph rendering and mouse interaction. Mouse click event creates a popup window which will eventually render a single bicluster using specified layout

```
package com.bb.graph;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.util.ArrayList;
import javax.swing.JFrame;
import javax.swing.JInternalFrame;
import javax.swing.JOptionPane;

/**
 *
 * @author mohammad bozlulkarim
 */
public class frmBipartitegraph extends javax.swing.JPanel {

    /**
     * Creates new form frmBipartitegraph
     */
    private myClusterbi  mbcls= new myClusterbi();
    private  Color bcolor;
    private int x;
    private int y;
    private int mz=0;
    private String sk="";

    public frmBipartitegraph() {
        initComponents();
    }
    public frmBipartitegraph(myClusterbi mb , Color col) {
        initComponents();
        mbcls = mb;
        bcolor = clsWindow.fbackcolor;
    }

    @Override
    public void paintComponent(Graphics g) {
     setBackground(bcolor);
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D) g;
    g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,RenderingHints.VALUE_TEXT_ANTI
ALIAS_ON);
    graphColCor temp ,temp1, to;

//        g2d.drawString("test", 200, 200);
```

```
      ArrayList adlist;
      for(String key: mbcls.nodeAbt.keySet()){
        temp = (graphColCor) mbcls.nodeAbt.get(key);
        g2d.setColor(temp.Col);
        g2d.fillOval(temp.xpos-5, temp.ypos-5, 10, 10);
        g2d.drawOval(temp.xpos-5, temp.ypos-5, 10, 10);

        g2d.setColor(Color.BLACK);
        g2d.setFont(clsWindow.vfont);
        g2d.drawString(key, temp.xpos-2, temp.ypos-6);
        adlist = (ArrayList) mbcls.adlistf.get(key);
        if (adlist != null){
          for (int i = 0; i < adlist.size(); i++) {
            String  adkey =(String) adlist.get(i);
            try {
             to = (graphColCor)  mbcls.nodeAbt.get(adkey);
             float alpha = (float) 0.4;
            Color color = new Color(0, 0, 0, alpha); //Red
             //g2d.setPaint(clsWindow.edgcol);
           //  g2d.setPaint(edcol);
             //g2d.setColor(Color.BLACK);
             g2d.setColor(clsWindow.edgcol);

             g2d.drawLine(temp.xpos, temp.ypos, to.xpos ,to.ypos );

            } catch (Exception e) {
            }
          }

          }


      }


    for(int i=0 ; i<=mbcls.frmregion.size()-1; ++i)
    {
         region r = (region) mbcls.frmregion.get(i);
       //  g2d.setColor(Color.BLACK);
          g2d.drawRect(r.xul,r.yul,r.xlr-r.xul,r.ylr-r.yul);
    }

    g2d.dispose();
  // clsWindow.bicluster = this;
   clsWindow.clusterform = this;
  }


    private void formMouseClicked(java.awt.event.MouseEvent evt) {
        if (evt.getClickCount() == 1) {
          return;
         }
        int rpos;
        rpos = this.mbcls.findregion(x, y);

          try {

        JInternalFrame frame = new JInternalFrame("Single Bicluster, Bicluster No " +
mbcls.frmregionkey.get(rpos), true,true,false,false );
        frame.setClosable(true);
        //myClusterbi smbb = new myClusterbi(mbcls);
        frmBiclusterdrawsub m1 = new frmBiclusterdrawsub(mbcls, rpos);
        frame.setBounds(0, 0, 800,400 );
        clsAllForm.fmain.makeinternal(frame);
        frame.getContentPane().add(m1);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setVisible(true);
        } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Click Valid Region");
        }


    }

    private void formMousePressed(java.awt.event.MouseEvent evt) {

      x = evt.getX();
      y = evt.getY();
      sk = this.mbcls.selectnode(x, y);
// TODO add your handling code here:
    }

    private void formMouseDragged(java.awt.event.MouseEvent evt) {
       int dx = evt.getX() - x;
      int dy = evt.getY() - y;
    //  this.clsr.maingraphd.changenodecoor(x, y, dx, dy);
      this.mbcls.changenodecoorsel(sk, dx, dy);

      this.repaint();
      x += dx;
      y += dy;
        // TODO add your handling code here:
    }


    // Variables declaration - do not modify
    // End of variables declaration
}


/*
 * DPClusSBO1.1 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * DPClusSBO1.1 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */

package com.bb.graph;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.AffineTransform;
import java.util.ArrayList;
import java.util.HashMap;

/**
 *
 * @author hira
 */
```

This class renders bicluster set by using a grid layout. No mouse interaction is incorporated.

```
public class frmBIclusterdraw extends javax.swing.JPanel {

   /**
    * Creates new form frmBIclusterdraw
    */
    clsMatrixData mdata;
    clsBicluster bidata ;
    int bvalue[][];

   public frmBIclusterdraw(clsMatrixData m, clsBicluster b) {
      initComponents();
      //this.setPreferredSize(5000,5000);
      mdata = m;
      bidata = b;
      bidata.setdata();
     // bidata.setdatarandom();
      bvalue = new int[bidata.cft.size()][bidata.cst.size()];

    for(int i=0; i<=bidata.cft.size()-1; ++i)
    {
      String s= bidata.cft.get(i);

      for(int j=0; j<=bidata.cst.size()-1; ++j)
       {
       String r= bidata.cst.get(j);
       if (mdata.getmatdata(s, r)==1)
         bvalue[i][j]=1;
       else  bvalue[i][j]=0;
       }
    }

   }

   @Override

   public void paintComponent(Graphics g) {
     setBackground(clsWindow.fbackcolor);
     super.paintComponent(g);
   //JOptionPane.showMessageDialog(null, "Enter valid Density");

   Graphics2D g2d = (Graphics2D) g;
   g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
   g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
   RenderingHints.VALUE_TEXT_ANTIALIAS_ON);

   String s,r;
   int fsize =0;
   fsize = clsWindow.vfont.getSize();
   Font fontf = new Font(null, Font.PLAIN, fsize);

   for(int i=0 ; i<=bidata.cft.size()-1; ++i)
    {
      g2d.setColor(Color.BLACK);
      g2d.setFont(fontf);
      s= bidata.cft.get(i);
      g2d.drawString( s ,300- g2d.getFontMetrics().stringWidth(s) , 310+i*fsize );
   }

   Font font = new Font(null, Font.PLAIN, fsize);
   AffineTransform affineTransform = new AffineTransform();
   affineTransform.rotate(Math.toRadians(-90), 0, 0);
   Font rotatedFont = font.deriveFont(affineTransform);
   g2d.setFont(rotatedFont);

   for(int i=0 ; i<=bidata.cst.size()-1; ++i)
   {
      s= bidata.cst.get(i);
      g2d.drawString( s ,310+i*fsize, 300 );

   }
    g2d.setColor(Color.GREEN);

    for(int i=0; i<=bidata.cft.size()-1; ++i)
    {
      for(int j=0; j<=bidata.cst.size()-1; ++j)
      {
       if (bvalue[i][j]==1)
        {
         g2d.fillOval(310+j*fsize-2, 310+i*fsize-2, 4, 4);
         g2d.drawOval(310+j*fsize-2, 310+i*fsize-2, 4, 4);
        }
      }
    }

    g2d.dispose();
    //clsWindow.bicluster = this;
    clsWindow.clusterform = this;
   }



   // Variables declaration - do not modify
   // End of variables declaration      }

/*
 * DPClusSBO1.1 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * DPClusSBO1.1 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */
```

This class is used to create a hierarchical graph of both simple clusters and biclusters. Node radius and edges width are calculated using cluster size and the common number of vertices.

```
package com.bb.graph;

import com.sun.javafx.scene.layout.region.Margins;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.Vector;
import java.util.concurrent.ThreadLocalRandom;
import javax.crypto.Mac;

/**
 *
 * @author hira
 */
public class graphCluster {
   public  HashMap<String,nodeCluster> grpclus ;
   private Vector clsdata;
   public Hashtable adlist;
   private ArrayList<Integer> clusterno;
   private  ArrayList<String> ovnode;
   private  ArrayList<String> ovnodes;

   private clsBicluster bicls = new clsBicluster()  ;
   private int ovsize;
   public Hashtable getAdlist() {
      return adlist;
   }



    public graphCluster(Vector clsdata, Hashtable adlist , ArrayList cluno)  {
       try {
       this.clsdata =  (Vector) clsWindow.deepCopy(clsdata) ;
       this.adlist = (Hashtable) clsWindow.deepCopy(adlist);
       this.clusterno = (ArrayList) clsWindow.deepCopy(cluno);
       } catch (Exception e) {
       }
       grpclus = new HashMap<>();
   }

    public graphCluster(Vector clsdata, Hashtable adlist , ArrayList cluno, clsBicluster camat)  {
       try {
       this.clsdata =  (Vector) clsWindow.deepCopy(clsdata) ;
       this.adlist = (Hashtable) clsWindow.deepCopy(adlist);
       this.clusterno = (ArrayList) clsWindow.deepCopy(cluno);
       this.bicls  =  (clsBicluster) (ObjectCloner.deepCopy(camat));
// this.bicls = (clsBicluster) clsWindow.deepCopy(camat);

       } catch (Exception e) {
       }
      // this.bicls = camat;
       grpclus = new HashMap<>();
   }

   private int edgebcluster( Vector first , Vector Second )
   {
      Vector adv;
      int count=0;
      for(int i=0; i<= first.size()-1; i++){
          if ( Second.contains(first.elementAt(i))) continue;
         adv=(Vector)adlist.get( first.elementAt(i) );
           for(int j=0; j<= adv.size()-1; j++){
               if  (Second.contains(adv.elementAt(j)))
                  count++;
           }

      }
   return count ;
   }

      private int edgebcluster2( Vector first , Vector Second )
```

```
{
    Vector adv;
    int count=0;
    for(int i=0; i<= first.size()-1; i++){
        if ( Second.contains(first.elementAt(i))) count++;

    }
return count ;
}


    private int edgebcluster1( Vector first1 , Vector Second1 )
{
    Vector adv , dup  ;
    int count=0;
    Vector first, Second;
    first = new Vector();
    Second = new Vector();

    first.addAll(0,first1);
    Second.addAll(0,Second1);


    dup = new Vector();
    for(int p=0; p<= first.size()-1; p++){
      if ( Second.contains(first.elementAt(p)))
      {
          dup.add(first.get(p));
        if ( ovnode.contains(first.get(p))==false )
        { ovnode.add((String) first.get(p)); }
      }
    }

   if (dup != null)
   {
    first.removeAll(dup);
    Second.removeAll(dup);
   }
    for(int i=0; i<= first.size()-1; i++){
      adv=(Vector)adlist.get( first.elementAt(i) );
        for(int j=0; j<= adv.size()-1; j++){
            if  (Second.contains(adv.elementAt(j)))
              count++;
        }

    }
return count ;
}


public void setcoordinate( int xl, int yl  )
{
   int p1 =0, p2 =0;
    float radius = 150;
   int clkey[] = new int[grpclus.size()];
  float a = (float) (xl-70)/2;
  float b = (float) (yl-70)/2;
  int   k=0 ,p;
    float angle =0 ;
  int i=0;
  String ss= "";
  for (String key : grpclus.keySet()) {
    ss = key.substring(1, key.length());
    clkey[i] = Integer.parseInt(key.substring(1, key.length()));
```

```
        ++i;
      }
       Arrays.sort(clkey);
     i=0;
    for (String key : grpclus.keySet()) {
        if (clsWindow.grpview ==1)
          {
          p1 =ThreadLocalRandom.current().nextInt(50, xl-50);
          p2 = ThreadLocalRandom.current().nextInt(50, yl-50);
         }
         else if (clsWindow.grpview ==2)
         {
          angle = (float) ((2*Math.PI/grpclus.size())*(k+1));
          ++k;
        //   p1 = (int )( xl/2+ (a-20)*Math.cos(angle));
        //   p2 = (int )( yl/2+ (b-20)*Math.sin(angle));
          p1 = (int )( xl/2+ a*Math.cos(angle));
          p2 = (int )( yl/2+ b*Math.sin(angle));

         }

        else if (clsWindow.grpview ==3)
         {
          angle = (float) ((2*Math.PI/grpclus.size())*(k+1));
          ++k;
          p =ThreadLocalRandom.current().nextInt(0, 2);

         if (p==1)
          {
          p1 = (int )( xl/2+ (a-100)*Math.cos(angle));
          p2 = (int )( yl/2+ (b-100)*Math.sin(angle));
          }
         else
          {
          p1 = (int )( xl/2+ a*Math.cos(angle));
          p2 = (int )( yl/2+ b*Math.sin(angle));
          }
         }
       ss = "C" + Integer.toString(clkey[i]) ;
       grpclus.get(ss).xcor = p1;
       grpclus.get(ss).ycor = p2;
        ++i;
      }


    }

public String selectnode(int x , int y)
{
  String sk="";
  for (String key : grpclus.keySet()) {
          nodeCluster ns = grpclus.get(key);
       if (((x > ns.xcor-15) && (x < ns.xcor+15)) && ((y > ns.ycor-15) && (y < ns.ycor+15)))
       {
           sk = key;
           break;
       }
  }

  return sk;
}

public void changenodecoorsel(String sk, int dx , int dy)
{
     if   (grpclus.get(sk) != null)
     {
      grpclus.get(sk).xcor = grpclus.get(sk).xcor +dx;
      grpclus.get(sk).ycor = grpclus.get(sk).ycor +dy;
     }

}

public void changenodecoor(int x , int y , int dx , int dy)
{


  for (String key : grpclus.keySet()) {
          nodeCluster ns = grpclus.get(key);
      if (((x > ns.xcor-15) && (x < ns.xcor+15)) && ((y > ns.ycor-15) && (y < ns.ycor+15)))
      {
         grpclus.get(key).xcor = ns.xcor +dx;
         grpclus.get(key).ycor = ns.ycor +dy;
           break;
      }

     }

}


    public void setClusterdata()
    {
        Vector clusf, cluss;
        nodeCluster ncls;
        String clname ="";
        HashMap<String, Integer> adclus,adclussize ;

        int clsize = 0 , clsizead = 0, comedge= 0;
        for(int i=0; i<= clsdata.size()-1; i++){
             clusf=(Vector) clsdata.elementAt(i);
             //clname = "C"+  Integer.toString(i+1);
             clname = "C"+  clusterno.get(i).toString();
             ovnode = new ArrayList<>();
             clsize = (int) (Math.log(clusf.size()+1)*10 / Math.log(2));
            adclus = new HashMap<>();
            adclussize  = new HashMap<>();
             for(int j=0; j<= clsdata.size()-1; j++){
                  if (i==j) continue;
               cluss=(Vector) clsdata.elementAt(j);
                comedge = edgebcluster1(clusf, cluss);
              if (comedge>0)
              {
             //   adclus.put("C"+  Integer.toString(j+1), comedge);
                adclus.put("C"+ clusterno.get(j).toString()   , comedge);
                clsizead = (int) (Math.log(cluss.size()+1)*10 / Math.log(2));

                adclussize.put("C"+ clusterno.get(j).toString()   , clsizead);

              }

            }
            ncls = new nodeCluster();
            ncls.setAdclus(adclus);
            ncls.setAdcluscsize(adclussize);
            ncls.setCname(clname);
            ncls.setCsize(clsize);
            ncls.setMslist(Collections.list(clusf.elements()));
            ncls.setMsovlist(ovnode);
            grpclus.put(clname, ncls);

        }


    }

   private int relationbiclus( ArrayList first , ArrayList Second , ArrayList f1 , ArrayList s1 )
   {
      double per;
     ovsize =0;
     ArrayList <String> thr = new ArrayList<>(first);
     thr.retainAll(Second);
     per = ((double) thr.size()/(double) (first.size()+Second.size()-thr.size())) *100 ;
    // ovnode = new ArrayList<>(f1);
    // ovnode.retainAll(s1);
      ovnode = new ArrayList<>(first);
      ovnode.retainAll(Second);

    return (int) per;
   }




    public void setClusterdatabi()
    {
        Vector clusf, cluss;
        nodeCluster ncls;
        String clname ="";
        HashMap<String, Integer> adclus,adclussize ;
        int hjcf;
        hjcf = Integer.parseInt(clsWindow.hjoincoff);
        ArrayList<Integer> keys;
        keys = new ArrayList<Integer>(bicls.getBclus().keySet());
        ArrayList<String> std1 ;
        ArrayList<String> std2 ;
        ArrayList<String> ftd1 ;
        ArrayList<String> ftd2 ;

        int clsize = 0 , clsizead = 0, comedge= 0;

     for(int i=0; i<=keys.size()-1; ++i )
      {
        ftd1  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(i))).getFtuple());
        std1  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(i))).getStuple());

        clname = "C"+  (keys.get(i)).toString();
        clsize = (int) (Math.log(ftd1.size()+1)*10 / Math.log(2));
        adclus = new HashMap<>();
        adclussize  = new HashMap<>();
        for(int j=0; j<= keys.size()-1; j++){
              if (i==j) continue;

         ftd2  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(j))).getFtuple());
         std2  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(j))).getStuple());
             comedge = relationbiclus(std1, std2 , ftd1, ftd2);

             if (comedge>hjcf)
              {
                //adclus.put("C"+ (keys.get(j)).toString()  , comedge);
                adclus.put("C"+ (keys.get(j)).toString()  , ovnode.size());

                clsizead = (int) (Math.log(ftd2.size()+1)*10 / Math.log(2));
```

```
              adclussize.put("C"+ (keys.get(j)).toString()   , clsizead);

            }


        }
          ncls = new nodeCluster();
           ncls.setAdclus(adclus);
           ncls.setAdcluscsize(adclussize);
           ncls.setCname(clname);
           ncls.setCsize(clsize);
           ncls.setMslist(ftd1);
           ncls.setMsovlist(ovnode);
           grpclus.put(clname, ncls);
            }

    }

  private int relationbiclusdual( ArrayList first , ArrayList Second , ArrayList f1 , ArrayList s1 )
   {
       double per;
     ovsize =0;
     ArrayList <String> thr = new ArrayList<>(first);
     thr.retainAll(Second);
     ArrayList <String> thrs = new ArrayList<>(f1);
     thrs.retainAll(s1);

     per = ((double) (thr.size()*thrs.size())/(double) (first.size()*f1.size()+Second.size()*s1.size()-
thr.size()*thrs.size())) *100 ;
    // ovnode = new ArrayList<>(f1);
    // ovnode.retainAll(s1);
      ovnode = new ArrayList<>(first);
      ovnode.retainAll(Second);
      ovnodes = new ArrayList<>(f1);
      ovnodes.retainAll(s1);


    return (int) per;
   }

    public void setClusterdatabitwo()
    {
        Vector clusf, cluss;
        nodeCluster ncls;
        String clname ="";
        HashMap<String, Integer> adclus,adclussize ;
        int hjcf;
        hjcf = Integer.parseInt(clsWindow.hjoincoff);
        ArrayList<Integer> keys;
        keys = new ArrayList<Integer>(bicls.getBclus().keySet());
       ArrayList<String> std1 ;
       ArrayList<String> std2 ;
       ArrayList<String> ftd1 ;
       ArrayList<String> ftd2 ;

       int clsize = 0 , clsizead = 0, comedge= 0;

     for(int i=0; i<=keys.size()-1; ++i )
      {
        ftd1  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(i))).getFtuple());
        std1  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(i))).getStuple());

        clname = "C"+  (keys.get(i)).toString();
   clsize = (int) (Math.log(ftd1.size()+1)*Math.log(std1.size()+1)*3 / Math.log(2));
   adclus = new HashMap<>();
   adclussize  = new HashMap<>();
   for(int j=0; j<= keys.size()-1; j++){
        if (i==j) continue;

    ftd2  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(j))).getFtuple());
    std2  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(j))).getStuple());
       comedge = relationbiclusdual(std1, std2 , ftd1, ftd2);

       if (comedge>hjcf)
        {
          //adclus.put("C"+ (keys.get(j)).toString()  , comedge);
          double ls =  Math.sqrt(ovnode.size()*ovnodes.size());

          adclus.put("C"+ (keys.get(j)).toString()  ,(int) ls);

          clsizead = (int) (Math.log(ftd2.size()+1)*Math.log(std2.size()+1)*3 / Math.log(2));

          adclussize.put("C"+ (keys.get(j)).toString()   , clsizead);

        }


   }
      ncls = new nodeCluster();
       ncls.setAdclus(adclus);
       ncls.setAdcluscsize(adclussize);
       ncls.setCname(clname);
       ncls.setCsize(clsize);
       ncls.setMslist(ftd1);
       ncls.setMsovlist(ovnode);
       grpclus.put(clname, ncls);
       }

}


public void setClusterdatabione()
{
   Vector clusf, cluss;
   nodeCluster ncls;
   String clname ="";
   HashMap<String, Integer> adclus,adclussize ;
   int hjcf;
   hjcf = Integer.parseInt(clsWindow.hjoincoff);
   ArrayList<Integer> keys;
   keys = new ArrayList<Integer>(bicls.getBclus().keySet());
  ArrayList<String> std1 ;
  ArrayList<String> std2 ;
  ArrayList<String> ftd1 ;
  ArrayList<String> ftd2 ;

  int clsize = 0 , clsizead = 0, comedge= 0;

 for(int i=0; i<=keys.size()-1; ++i )
  {
   ftd1  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(i))).getFtuple());
   std1  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(i))).getStuple());

   clname = "C"+  (keys.get(i)).toString();
   clsize = (int) (Math.log(ftd1.size()+1)*10 / Math.log(2));
   adclus = new HashMap<>();
   adclussize  = new HashMap<>();
```

```
        for(int j=0; j<= keys.size()-1; j++){
             if (i==j) continue;

         ftd2  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(j))).getFtuple());
         std2  = new ArrayList<String>(((clsBinode) bicls.getBclus().get(keys.get(j))).getStuple());
            comedge = relationbiclus(ftd1, ftd2,std1, std2 );

            if (comedge>hjcf)
            {
               //adclus.put("C"+ (keys.get(j)).toString()  , comedge);
               adclus.put("C"+ (keys.get(j)).toString()  , ovnode.size());

               clsizead = (int) (Math.log(ftd2.size()+1)*10 / Math.log(2));

               adclussize.put("C"+ (keys.get(j)).toString()   , clsizead);

            }


        }
           ncls = new nodeCluster();
            ncls.setAdclus(adclus);
            ncls.setAdcluscsize(adclussize);
            ncls.setCname(clname);
            ncls.setCsize(clsize);
            ncls.setMslist(std1);
            ncls.setMsovlist(ovnode);
            grpclus.put(clname, ncls);
            }

    }


}
```

```
/*
 * DPClusSBO1.1 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * DPClusSBO1.1 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */
```

This class is used to render hierarchical graph plotting for bicluster with mouse interaction . A single node click will expand a popup window where hierarchical relation will be shown

```
package com.bb.graph;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import javax.swing.JFrame;
import javax.swing.JInternalFrame;
```

```
/**
 *
 * @author mohammad bozlulkarim
 */
public class frmClusterHierarchyBi extends javax.swing.JPanel {

    private graphCluster grd ;
    private HashMap<String, ArrayList<String>> nclAdlist;
    private clsMatrixData mt;
    private clsBicluster bcls;
    private myClusterhr mch;
    private int x;
    private int y;
    private String sk="";
    private  Color bcolor;
   /**
    * Creates new form frmClusterHierarchyBi
    */
   public frmClusterHierarchyBi() {
      initComponents();
   }

    public frmClusterHierarchyBi( graphCluster gl , clsMatrixData mt , clsBicluster bcl) {
      initComponents();
      this.grd = gl;
      this.mt = mt;
      this.bcls = bcl;
      bcolor = clsWindow.fbackcolor;

  }

  @Override
   public void paintComponent(Graphics g) {
   setBackground(bcolor);

   super.paintComponent(g);

   Graphics2D g2d = (Graphics2D) g;
   g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
   g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
   RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
    float lineth =0;

  for (String key : grd.grpclus.keySet())
   {
     nodeCluster ns = grd.grpclus.get(key);
       g2d.setColor(Color.BLUE);
    //  g2d.setColor(clsWindow.unnc);

      lineth =(float) 0.1;
      g2d.setStroke(new BasicStroke(lineth));

     g2d.fillOval(ns.xcor - ns.getCsize()/2 , ns.ycor-ns.getCsize()/2,  ns.getCsize(), ns.getCsize());
      g2d.drawOval(ns.xcor - ns.getCsize()/2 , ns.ycor-ns.getCsize()/2,  ns.getCsize(), ns.getCsize());

      g2d.setColor(Color.BLACK);
      g2d.setFont(clsWindow.vfont);
      g2d.drawString(ns.getCname(), ns.xcor + ns.getCsize()/2+2 , ns.ycor);
     Iterator it = ns.getAdclus().entrySet().iterator();
    while (it.hasNext()) {
        Map.Entry pair = (Map.Entry)it.next();
      nodeCluster nt = grd.grpclus.get(pair.getKey());
    //    lineth = (float) (Math.log((int) pair.getValue())/Math.log(2));
    // lineth =  (float) ((int) pair.getValue());
      lineth = (float) (Math.log((int) pair.getValue()+1)/Math.log(2));
      // g2d.setColor(Color.BLACK);
      if (lineth>0)
      {
      g2d.setColor(clsWindow.edgcol);
       g2d.setStroke(new BasicStroke(lineth));
      g2d.drawLine(ns.xcor, ns.ycor, nt.xcor,nt.ycor );
      }

  }




   }

    g2d.dispose();
     clsWindow.Hierarchyform = this;
   }

   private void formMousePressed(java.awt.event.MouseEvent evt) {
    x = evt.getX();
    y = evt.getY();
    sk =  this.grd.selectnode(x, y);
      // TODO add your handling code here:
   }

   private void formMouseDragged(java.awt.event.MouseEvent evt) {
    int dx = evt.getX() - x;
    int dy = evt.getY() - y;
    // this.grd.changenodecoor(x, y, dx, dy);
    this.grd.changenodecoorsel(sk, dx, dy);
    repaint();

    x += dx;
    y += dy;
      // TODO add your handling code here:
   }

   private void formMouseClicked(java.awt.event.MouseEvent evt) {

      if (evt.getClickCount() == 1) {
        return;
       }

      String  slcl =  this.grd.selectnode(x, y);
      myClusterhr  mch = new myClusterhr((nodeCluster) grd.grpclus.get(slcl), mt, bcls);

      JInternalFrame frame = new JInternalFrame("Single Bicluster, Bicluster No " + slcl,
true,true,false,false );
      frame.setClosable(true);
      //myClusterbi smbb = new myClusterbi(mbcls);
      try {
      frmClusterHierarchyBiSub m1 = new frmClusterHierarchyBiSub(mch);
      frame.setBounds(0, 0, 800,400 );
      clsAllForm.fmain.makeinternal(frame);
      frame.getContentPane().add(m1);
      frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      frame.setVisible(true);

      } catch (Exception e) {
      }
```

```
        // TODO add your handling code here:
    }


    // Variables declaration - do not modify
    // End of variables declaration
}

/*
 * DPClusSBO1.1 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * DPClusSBO1.1 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License
 * along with DPClusSBO1.1.  If not, see <https://www.gnu.org/licenses/>.
 */

package com.bb.graph;

import com.sun.java.swing.plaf.windows.resources.windows;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.AffineTransform;
import java.io.Serializable;
import java.util.ArrayList;

/**
 *
 * @author mohammad bozlulkarim
 */
```

Hierarchical relationship of a single cluster is drawn using this class

```
public class frmClusterHierarchyBiSub extends javax.swing.JPanel implements Serializable{

    /**
     * Creates new form frmClusterHierarchyBiSub
     */
    private myClusterhr clhr = new myClusterhr();
     private int x;
     private int y;
     private String sk="";
     private  Color bcolor;
    public frmClusterHierarchyBiSub() {
        initComponents();
    }
    public frmClusterHierarchyBiSub(myClusterhr mc) throws Exception {
        initComponents();

        this.clhr = (myClusterhr) ObjectCloner.deepCopy(mc);
        this.clhr.setcoordinate();
        this.bcolor = clsWindow.fbackcolor;

    }

    @Override
```

```
  public void paintComponent(Graphics g) {
  super.paintComponent(g);
  setBackground(bcolor);
  Graphics2D g2d = (Graphics2D) g;
  g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
  g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
  RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
   graphColCor temp, to;
  AffineTransform orig = g2d.getTransform();
  ArrayList<String> adlist;
  int csize;
  float lineth = (float) 0.3;

  for(String key : clhr.getAdlistf().keySet())
  {
     temp = (graphColCor) clhr.getNodeAbt().get(key);
      g2d.setColor(temp.Col);
      g2d.fillOval(temp.xpos-5, temp.ypos-5, 10, 10);
      g2d.drawOval(temp.xpos-5, temp.ypos-5, 10, 10);

      g2d.translate(temp.xpos, temp.ypos);
      g2d.rotate(-Math.PI/4);
      g2d.translate(-temp.xpos, -temp.ypos);
      g2d.setColor(Color.BLACK);
      g2d.setFont(clsWindow.vfont);
      g2d.drawString(key, temp.xpos+8, temp.ypos-4);
      g2d.setTransform(orig);

  }
  for(String key : clhr.getAdlists().keySet())
  {
     temp = (graphColCor) clhr.getNodeAbt().get(key);
      g2d.setColor(temp.Col);
      g2d.fillOval(temp.xpos-5, temp.ypos-5, 10, 10);
      g2d.drawOval(temp.xpos-5, temp.ypos-5, 10, 10);

      g2d.translate(temp.xpos, temp.ypos);
      g2d.rotate(Math.PI/4);
      g2d.translate(-temp.xpos, -temp.ypos);
      g2d.setColor(Color.BLACK);
      g2d.setFont(clsWindow.vfont);
      g2d.drawString(key, temp.xpos+6, temp.ypos+14);
      g2d.setTransform(orig);
      adlist = (ArrayList) clhr.getAdlists().get(key);

      for(int j=0; j<= adlist.size()-1; j++){
          String  adkey =(String) adlist.get(j);
           to = (graphColCor)  clhr.getNodeAbt().get(adkey);
           //g2d.setColor(Color.BLACK);
           g2d.setColor(clsWindow.edgcol);
           g2d.drawLine(temp.xpos, temp.ypos, to.xpos ,to.ypos );

      }

  }

  for(String key : clhr.getHnode().keySet())
  {
    temp = (graphColCor) clhr.getNodeAbt().get(key);
    csize = clhr.getHnode().get(key);
    g2d.setColor(temp.Col);
    g2d.fillOval(temp.xpos - csize/2 , temp.ypos-csize/2,  csize, csize);
    g2d.drawOval(temp.xpos - csize/2 , temp.ypos-csize/2,  csize, csize);
    g2d.setFont(clsWindow.vfont);
    g2d.drawString(key, temp.xpos + csize/2+4 ,temp.ypos);

    adlist = (ArrayList) clhr.getHlink().get(key);

      for(int j=0; j<= adlist.size()-1; j++){
          String  adkey =(String) adlist.get(j);
           to = (graphColCor)  clhr.getNodeAbt().get(adkey);
           g2d.setColor(Color.RED);
           g2d.setStroke(new BasicStroke(lineth));
           g2d.drawLine(temp.xpos, temp.ypos, to.xpos ,to.ypos );
           g2d.setColor(Color.RED);
           g2d.fillOval(to.xpos-5, to.ypos-5, 10, 10);
           g2d.drawOval(to.xpos-5, to.ypos-5, 10, 10);


      }


  }




  g2d.dispose();
  //clsWindow.Hierarchyform = this;
  clsWindow.Hierarchyformsub = this;
  }

  private void formMousePressed(java.awt.event.MouseEvent evt) {
    x = evt.getX();
    y = evt.getY();
    sk =  clhr.selectnode(x, y);
      // TODO add your handling code here:
  }

  private void formMouseDragged(java.awt.event.MouseEvent evt) {

    int dx = evt.getX() - x;
    int dy = evt.getY() - y;
    if (sk!="")
    clhr.changenodecoorsel(sk, dx, dy);
    repaint();

    x += dx;
    y += dy;
      // TODO add your handling code here:
  }


  // Variables declaration - do not modify
  // End of variables declaration
}
```

R Code

This R code is used to measure the performance of BiClusO to five different algorithms by creating synthetic data.

(Only one example of synthetic data and biological data is also shown)

Creating synthetic data by adding error to embedded bicluster set and applying different algorithm to generate biclusters

```
########################################### Cluster create
################################
########################################### Equal size ################################
########################################### Non overlapping
################################

sp <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\alias\\Env\\artifical data.xlsx",
sheet="Sheet1" ,startRow = 1, colNames = TRUE , rowNames = TRUE))
temp  <- sp


out <- data.frame(sl = numeric(), xcor =numeric() , ycor = numeric() )

sl = as.integer(5)
sl <- 0
for(i in 1:nrow(sp)) {

  for(j in 1:ncol(sp)) {
    if (sp[i,j]==0)
    {
      sl <- sl+1
      sss<- data.frame(sl,i,j)
      out <- rbind(out,sss)
    }

  }

}
b<-0.00

for(i in 1:9 ){

  b<- b + 0.01
  sp <- temp
  a <- as.integer(5)
  a <-  nrow(out)*b
  x <- as.integer( floor(runif(a, 1, nrow(out))))

  for(i in 1:length(x)) {
    sp[out[x[i],2],out[x[i],3]] <- 1
  }

  fpath <- paste("D:\\Bi cluster compare\\alias\\Env\\artifical data rend_",b,"",'.csv',sep='')

  write.table(sp,file=fpath, append=TRUE,sep=",",col.names=NA,row.names=TRUE)
}

b<-0.00

for(i in 1:9 ){
  b<- b + 0.01
  fpath <- paste("D:\\Bi cluster compare\\alias\\Env\\artifical data rend_",b,"",'.csv',sep='')

  sp <- as.data.frame(read.csv(fpath, row.names = 1, header= TRUE))
  mt <- as.matrix(sp)

      #ctype <- 'plaid'
      #res <- biclust(mt, method=BCPlaid(), verbose=FALSE)

      #ctype <- 'spectral'
      #res <- biclust(mt, method=BCSpectral(), normalization="log", numberOfEigenvalues=1,minr=2, minc=2,
withinVar=1)
      #ctype <- 'bimax'
      #res <- biclust(mt, method=BCBimax(), minr=2, minc=2)
      #ctype <- 'xmotif'
      #res <- biclust(mt, method=BCXmotifs(), ns=10, nd=1000, sd=7, alpha=0.1, number=100)
      ctype <- 'cc'
      res <- biclust(mt, method=BCCC(), delta = .0000005, alpha=1.2, number=100)

    fpath <- paste("D:\\Bi cluster compare\\alias\\Env\\result_",ctype,b,"",'.txt',sep='')
    writeBiclusterResults(fpath, res,ctype, dimnames(mt)[1][[1]],
                   dimnames(mt)[2][[1]])

}
```

Comparing the generated bicluster to original bicluster

```
############################################ Cluster compare ################################
############################################ Equal size  #################################
############################################ Non overlappin ################################


sp <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\alias\\Env\\artifical data.xlsx",
sheet="orgcluster" ,startRow = 1, colNames = TRUE , rowNames = FALSE))

bclst <- c('plaid','spectral','bimax','xmotif','cc')


grp <- data.frame(err = numeric(9),  pl = numeric(9),sp = numeric(9),bm = numeric(9),xm = numeric(9) ,cc =
numeric(9),dp = numeric(9), stringsAsFactors=FALSE )


'plaid'
dfi <-1
for (lp in bclst) {
  dfi<- dfi+1
  bmethod <-lp
  b<-0.00
 for (kr in 1:9) {

  b<- b + 0.01
  fpath <- paste("D:\\Bi cluster compare\\alias\\Env\\result_",bmethod,b,"",'.txt',sep='')

  #fileName <- "D:\\Bi cluster compare\\Env\\result_BCbimax0.01.txt"
  fileName <- fpath

  conn <- file(fileName,open="r")
  linn <-readLines(conn)
  out <- data.frame(fn =character()  , sn =character() , clsno = numeric() , stringsAsFactors=FALSE)
  k = as.integer(3)
  k=1
  for (i in 2:length(linn)){
   if ((i%%3)==0)
   {
      temp <- linn[i]
      temp <- gsub("\\ ",",",temp)
      temp1 <- linn[i+1]
      temp1 <- gsub("\\ ",",",temp1)

      out = rbind.data.frame(out,data.frame(fn=as.character(temp), sn = as.character(temp1),
clsno=k,stringsAsFactors=FALSE))


      k <- k+1
    }
  }
  close(conn)

  s<-0.0
  s1<-0.0

  for(i in 1:nrow(sp)) {
    a1<- unlist(strsplit(sp[i,2], ","))
    b1<- unlist(strsplit(sp[i,3], ","))

    ratm <- 0.0
    ratm1 <- 0.0

    for(j in 1:nrow(out)) {
      a2<- unlist(strsplit(out[j,1], ","))
      b2<- unlist(strsplit(out[j,2], ","))

      p1 <- intersect(a1,a2)
      p2 <- union(a1,a2)
      rat <-  length(p1)/length(p2)
      if (ratm < rat) ratm <- rat

      p1 <- intersect(b1,b2)
      p2 <- union(b1,b2)
      rat <-  length(p1)/length(p2)
      if (ratm1 < rat) ratm1 <- rat

    }
    s<- s+ ratm
    s1<- s1+ ratm1

  }

  s<- s/10
  s1<- s1/10
  d <- sqrt(s*s1)
  grp[kr, dfi] <-d
  grp[kr,1] <-b
 }

}

##############################Dp Clus

for (k in 1:9) {
  out <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\alias\\Env\\result_DPClusBiCls.xlsx",
sheet=k ,startRow = 1, colNames = TRUE , rowNames = FALSE))
  s<-0.0
  s1<-0.0

  for(i in 1:nrow(sp)) {
    a1<- unlist(strsplit(sp[i,2], ","))
    b1<- unlist(strsplit(sp[i,3], ","))
    ratm <- 0.0
    ratm1 <- 0.0

    for(j in 1:nrow(out)) {
      a2<- unlist(strsplit(out[j,2], ","))
      b2<- unlist(strsplit(out[j,3], ","))

      p1 <- intersect(a1,a2)
      p2 <- union(a1,a2)
      rat <-  length(p1)/length(p2)
      if (ratm < rat) ratm <- rat

      p1 <- intersect(b1,b2)
      p2 <- union(b1,b2)
      rat <-  length(p1)/length(p2)
      if (ratm1 < rat) ratm1 <- rat

    }
    s<- s+ ratm
    s1<- s1+ ratm1

  }

  s<- s/10
  s1<- s1/10
  d <- sqrt(s*s1)
  grp[k, 7] <-d

}




plot(grp$dp, type = "o", col = "black", xlab = "Tanimoto coefficient", ylab = "Number of cluster",
     main = "(d) Edge number vs Tanimoto coefficient", ylim=c(0,1),xaxt="n", lwd=2)

lines(grp$pl,col="red", type = "o", lwd=3)
lines(grp$sp,col="green", type = "o", lwd=3)
lines(grp$bm,col="blue", type = "o", lwd=4)
lines(grp$xm,col="darkgreen", type = "o", lwd=3)
lines(grp$cc,col="deepskyblue4", type = "o", lwd=2)



plot(grp$dp, type = "o", col = "black", xlab = "Noise rate ", ylab = "Matching Score",
     main = "Equal size and non overlapping", ylim=c(0,1.2),xaxt = "n", cex=1.5 ,cex.axis=1.5, cex.lab = 1.5,
cex.main=1.7, lwd=2,pch = 0)

lines(grp$pl,col="red", type = "o",cex=1.5, lwd=2,pch = 1)
lines(grp$sp,col="green", type = "o",cex=1.5, lwd=2,pch = 2)
lines(grp$bm,col="blue", type = "o",cex=1.5, lwd=2,pch = 3)
lines(grp$xm,col="darkgreen", type = "o",cex=1.5, lwd=2,pch = 4)
lines(grp$cc,col="deepskyblue4", type = "o",cex=1.5, lwd=2,pch = 5)

axis(1, at=1:9, labels=paste0(".0", seq(1:9)),cex.axis=1.5)
```

Applying Fisher exact test to Species-VOC bipartite graph for different algorithm

```
install.packages('openxlsx')
install.packages('biclust')
```

```
library("openxlsx")
library("biclust")

dis <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\alias\\data.xlsx", sheet="adata" ,startRow = 1,
colNames = TRUE , rowNames = FALSE))
cls <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\alias\\data.xlsx",
sheet="clustdual5.033" ,startRow = 1, colNames = TRUE , rowNames = FALSE))


df <- subset(dis, select = c('sp','sclass'))
df <- df[!duplicated(df), ]
rownames(df) <-NULL

clsh <- data.frame( clsno = numeric(1000) , fn =character(1000) , cl=character(1000),
stringsAsFactors=FALSE)

lp <-0
for(i in 1:nrow(cls)) {
  a1<- unlist( strsplit(cls[i,2],',') )

  for(j in 1: length(a1)) {
    ch <- trimws(a1[j], "both")
    fsub <- df[df$sp == ch, ]

    lp<- lp+1
    clsh[lp,1]= i
    clsh[lp,2]= ch
    clsh[lp,3]= fsub[1,2]

    }
}

clsh <-clsh[clsh$clsno>0,]
dist <- table(clsh[,2])
nodeno <- length(dist)

pvalue <- data.frame( clsno = numeric() ,f1 =numeric() ,  f2 =numeric() ,f3 =numeric() ,f4 =numeric() , f5
=numeric() , f6 = character(), stringsAsFactors=FALSE)

#fsub <- clsh[clsh$clsno == 2, ]


for(i in 1:nrow(cls)) {
  fsub <- clsh[clsh$clsno == i, ]
  dist <- table(fsub[,3])
  p <-as.data.frame(dist)
  p  <- p[with(p, order(-p$Freq)), ]
  incls <- p[1,2]
  rincls <- sum(p[,2])-p[1,2]

  fsub <- clsh[clsh$clsno == i & clsh$cl==p[1,1], ]
  t1 <- unique(fsub$fn)

  fsub <- clsh[clsh$clsno != i & clsh$cl==p[1,1], ]
  t2 <- unique(fsub$fn)
  t <- setdiff(t2,t1)
  outcls <-length(t)

  Convictions <-
    matrix(c(incls, outcls, rincls, nodeno-outcls-rincls-incls),
         nrow = 2,
         dimnames =
           list(c("sc", "snc"),
                c("class", "rclass")))
  p<-fisher.test(Convictions,alternative = "greater")$p.value
  pvalue =   rbind(pvalue, data.frame(clsno=i, f1 = incls, f2 = outcls,f3 = rincls,f4 = nodeno-outcls-rincls-
incls,f5 = p, f6 ='dpcluso' ))

}
bidpclass <- pvalue
write.table(pvalue,file="D:\\Bi cluster compare\\alias\\pvalue.csv",
append=TRUE,sep=",",col.names=FALSE,row.names=FALSE)
nrow(bidpclass)
############################ Bi cluster from other algorithm

sp <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\alias\\spvoc.xlsx", sheet="Sheet1" ,startRow = 1,
rowNames = TRUE, colNames = TRUE))

pvalue <- data.frame( clsno = numeric() ,f1 =numeric() ,  f2 =numeric() ,f3 =numeric() ,f4 =numeric() , f5
=numeric() , f6 =character() , stringsAsFactors=FALSE)
dis <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\alias\\data.xlsx", sheet="adata" ,startRow = 1,
colNames = TRUE , rowNames = FALSE))

for (tr in 1:5) {

  mt <- as.matrix(sp)
  ctype <- ''
  res<-NULL
  if (tr == 1) {
    ctype <- 'plaid'
    res <- biclust(mt, method=BCPlaid(), verbose=FALSE)
   }
  if (tr == 2) {
    ctype <- 'spectral'
    res <- biclust(mt, method=BCSpectral(), normalization="log", numberOfEigenvalues=1,minr=2, minc=2,
withinVar=1)
  }
  if (tr == 3) {
    ctype <- 'bimax'
    res <- biclust(mt, method=BCBimax(), minr=2, minc=2)
  }
  if (tr == 4) {
    ctype <- 'xmotif'
    res <- biclust(mt, method=BCXmotifs(), ns=10, nd=1000, sd=7, alpha=0.1, number=100)
  }
  if (tr == 5) {
    ctype <- 'cc'
    res <- biclust(mt, method=BCCC(), delta = .0000005, alpha=1.2, number=100)
  }



fname <- paste('D:\\Bi cluster compare\\alias\\', ctype,'.txt', sep='')

writeBiclusterResults(fname, res,"biomax", dimnames(mt)[1][[1]],
                      dimnames(mt)[2][[1]])

}

for (tr in 1:5) {

  if (tr == 1) {
    ctype <- 'plaid'
  }
  if (tr == 2) {
    ctype <- 'spectral'
  }
  if (tr == 3) {
    ctype <- 'bimax'
  }
  if (tr == 4) {
    ctype <- 'xmotif'
  }
  if (tr == 5) {
    ctype <- 'cc'
  }
  fname <- paste('D:\\Bi cluster compare\\alias\\', ctype,'.txt', sep='')


conn <- file(fname,open="r")
linn <-readLines(conn)
out <- data.frame( clsno = numeric(), species =character() , stringsAsFactors=FALSE )
k = as.integer(3)
k=1
for (i in 2:length(linn)){
  if ((i%%3)==0)
  {
       temp <- unlist(strsplit(linn[i], ' '))
       out = rbind(out,data.frame(clsno=k,species=paste(temp,collapse=",")))
     k<-k+1
  }
}
close(conn)
nrow(out)
cls <-out
df <- subset(dis, select = c('sp','sclass'))
df <- df[!duplicated(df), ]
rownames(df) <-NULL

clsh <- data.frame( clsno = numeric(1000) , fn =character(1000) , cl=character(1000),
stringsAsFactors=FALSE)

lp <-0
for(i in 1:nrow(cls)) {

  a1<- unlist( strsplit(as.character(cls[i,2]),',') )

  for(j in 1: length(a1)) {
    ch <- trimws(a1[j], "both")
    fsub <- df[df$sp == ch, ]

    lp<- lp+1
    clsh[lp,1]= i
    clsh[lp,2]= ch
    clsh[lp,3]= fsub[1,2]

  }
}

clsh <-clsh[clsh$clsno>0,]
dist <- table(clsh[,2])
nodeno <- length(dist)


#fsub <- clsh[clsh$clsno == 2, ]


for(i in 1:nrow(cls)) {
  fsub <- clsh[clsh$clsno == i, ]
  dist <- table(fsub[,3])
  p <-as.data.frame(dist)
  p  <- p[with(p, order(-p$Freq)), ]
  incls <- p[1,2]
  rincls <- sum(p[,2])-p[1,2]

  fsub <- clsh[clsh$clsno == i & clsh$cl==p[1,1], ]
  t1 <- unique(fsub$fn)

  fsub <- clsh[clsh$clsno != i & clsh$cl==p[1,1], ]
  t2 <- unique(fsub$fn)
  t <- setdiff(t2,t1)
  outcls <-length(t)

  Convictions <-
    matrix(c(incls, outcls, rincls, nodeno-outcls-rincls-incls),
           nrow = 2,
           dimnames =
             list(c("sc", "snc"),
                  c("class", "rclass")))
  p<-fisher.test(Convictions,alternative = "greater")$p.value
  pvalue =   rbind(pvalue, data.frame(clsno=i, f1 = incls, f2 = outcls,f3 = rincls,f4 = nodeno-outcls-rincls-
incls,f5 = p, f6= ctype ))

}

}


write.table(pvalue,file="D:\\Bi cluster compare\\alias\\pvalue.csv",
append=TRUE,sep=",",col.names=FALSE,row.names=FALSE)
nrow(pvalue)

d1 <- rbind(subset(bidpclass, select = c('f5','f6')),subset(pvalue, select = c('f5','f6')))
typeof(d1$f5)
d1<- transform(d1 , f7= -log(d1$f5))
pp<-table(d1$f6)

d1 <- d1[d1$f5 < 0.05, ]
nrow(d1)
write.table(d1,file="D:\\Bi cluster compare\\alias\\pvalueor.csv",
append=TRUE,sep=",",col.names=FALSE,row.names=FALSE)

d <- (max(d1$f7)-min(d1$f7))/20


##################################
grp <- data.frame( dp = numeric(), pl = numeric(),sp = numeric(),bm = numeric(),xm = numeric() ,cc =
numeric(), stringsAsFactors=FALSE )

m <- max(d1$f7)
for(i in 1:20) {
  m <- m-d
  #print(m)
  dis <-d1[d1$f7 > m, ]
  dist <- table(dis$f6)
  p <-as.data.frame(dist)
  grp =   rbind(grp, data.frame(dp=p[1,2], pl = p[2,2], sp = p[3,2], bm = p[4,2],xm =p[5,2] ,cc = p[6,2] ))

}

######################################

d2 <- d1[with(d1, order(-f7)), ]
d2 <- cbind(d2, "f8"=1:nrow(d2))
nrow(d2)
```

```
grp <- data.frame( dp = numeric(), pl = numeric(),sp = numeric(),bm = numeric(),xm = numeric() ,cc =
numeric(), stringsAsFactors=FALSE )
m<-0
for(i in 1:17) {

  m <- i*20
  #print(m)
  dis <-d2[d2$f8 < m, ]
  dist <- table(dis$f6)
  p <-as.data.frame(dist)
  grp =   rbind(grp, data.frame(dp=p[1,2], pl = p[2,2], sp = p[3,2], bm = p[4,2],xm =p[5,2] ,cc = p[6,2] ))

}

#write.table(grp,file="D:\\Bi cluster compare\\alias\\grp.csv",
append=TRUE,sep=",",col.names=FALSE,row.names=FALSE)
#write.table(d1,file="D:\\Bi cluster compare\\alias\\d1.csv",
append=TRUE,sep=",",col.names=FALSE,row.names=FALSE)

plot(grp$dp, type = "o", col = "black", xlab = "Tanimoto coefficient", ylab = "Number of cluster",
     main = "(d) Edge number vs Tanimoto coefficient", ylim=c(0,150),xaxt="n", lwd=2)

lines(grp$pl,col="red", type = "o", lwd=3)
lines(grp$sp,col="green", type = "o", lwd=3)
lines(grp$bm,col="blue", type = "o", lwd=4)
lines(grp$xm,col="darkgreen", type = "o", lwd=3)
lines(grp$cc,col="deepskyblue4", type = "o", lwd=2)

rasters <- sprintf("Top %s",seq(10,340, by=20))

plot(grp$dp, type = "o", col = "black", xlab = "Ranking of biclusters", ylab = "Number of biclusters",
     main = "", ylim=c(0,150),xlim=c(1,17), cex=1.5 ,cex.axis=1.5, cex.lab = 1.5, lwd=2,pch = 0)

plot(grp$dp, type = "o", col = "black", xlab = "Ranking of biclusters", ylab = "Number of biclusters",
     main = "", ylim=c(0,150),xaxt = "n", cex=1.5 ,cex.axis=1.5, cex.lab = 1.5, lwd=2,pch = 0)
axis(1, at=1:17, labels=rasters,cex.axis=1.2,  cex.lab =2)

lines(grp$pl,col="red", type = "o",cex=1.5, lwd=2,pch = 1)
lines(grp$sp,col="green", type = "o",cex=1.5, lwd=2,pch = 2)
lines(grp$bm,col="blue", type = "o",cex=1.5, lwd=2,pch = 3)
lines(grp$xm,col="darkgreen", type = "o",cex=1.5, lwd=2,pch = 4)
lines(grp$cc,col="deepskyblue4", type = "o",cex=1.5, lwd=2,pch = 5)


ts.plot(time,gpars= list(col=rainbow(10)))
legend("topleft", legend = 1:10, col = 1:10, lty = 1)

legend('topleft', c("Health","Defense"))


################################  creating go term  p value ################

library("GOstats")
library("AnnotationDbi")
library("org.Sc.sgd.db")

frame = toTable(org.Sc.sgdGO)
goframeData = data.frame(frame$go_id, frame$Evidence, frame$systematic_name)
goFrame = GOFrame(goframeData, organism = "C. Cerevisiae")
head(goframeData)
goFrame = GOFrame(goframeData, organism = "CC")
goAllFrame = GOAllFrame(goFrame)

library("GSEABase")
```

```
gsc <- GeneSetCollection(goAllFrame, setType = GOCollection())


library("openxlsx")
sp <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\Paper Journal\\data.xlsx",
sheet="original" ,startRow = 1, rowNames = FALSE, colNames = TRUE))

unvg <- as.vector(sp[,1])

for(i in 1:nrow(tcls)) {
  print(i)
  clsg <- unlist( strsplit(tcls[i,2],',') )
  ##  BP MF CC
  params <- GSEAGOHyperGParams(name = "My Custom GSEA based annot Params", geneSetCollection =
gsc, geneIds = clsg, universeGeneIds = unvg, ontology = "MF", pvalueCutoff = 0.5, conditional = FALSE,
testDirection = "over")
  Over <- hyperGTest(params)
  s <- data.frame(summary(Over))
  maxvalu = s$Pvalue[1]
  tcls[i,4] <- maxvalu
  tcls[i,5] <- -log(maxvalu)
}
write.csv(tcls, file = "D:\\Bi cluster compare\\Paper Journal\\tclspval.csv", row.names = FALSE)

library(plyr)

spn <- as.data.frame(read.xlsx("D:\\Bi cluster compare\\Paper Journal\\data.xlsx", sheet="r-1.5-ov-0.25-
MF" ,startRow = 1, rowNames = FALSE, colNames = TRUE))
nrow(spn)

sp1 <- spn[spn$algo == "BiClusO", ]
sp1 <- head(arrange(sp1,desc(sp1$plog)), n = 53)
sp2 <- spn[spn$algo == "bimax", ]
sp2 <- head(arrange(sp2,desc(sp2$plog)), n = 53)
sp3 <- spn[spn$algo == "cc", ]
sp3 <- head(arrange(sp3,desc(sp3$plog)), n = 53)
sp4 <- spn[spn$algo == "plaid", ]
sp4 <- head(arrange(sp4,desc(sp4$plog)), n = 53)
sp5 <- spn[spn$algo == "xmotif", ]
sp5 <- head(arrange(sp5,desc(sp5$plog)), n = 53)

spn <- rbind(sp1,sp2,sp3,sp4,sp5)
dn2 <- spn[with(spn, order(-spn$plog)), ]

dn2 <- cbind(dn2, "f8"=1:nrow(dn2))
nrow(dn2)

grpn <- data.frame( dp = numeric(), bm = numeric(),cc = numeric(),pl = numeric() ,xm = numeric(),
stringsAsFactors=FALSE )
m<-0
for(i in 1:7) {

  m <- i*30
  #print(m)
  disn <-dn2[dn2$f8 < m, ]
  distn <- table(disn$algo)
  p <-as.data.frame(distn)
  grpn =   rbind(grpn, data.frame(dp=p[1,2], bm = p[2,2], cc = p[3,2],pl =p[4,2] ,xm = p[5,2] ))

}
rasters <- sprintf("Top %s",seq(30,210, by=30))
grp<-grpn
plot(grp$dp, type = "o", col = "black", xlab = "Ranking of biclusters", ylab = "Number of biclusters",
     main = "", ylim=c(0,60),xaxt = "n", cex=1.5 ,cex.axis=1.5, cex.lab = 1.5, lwd=2,pch = 0)
```

```
axis(1, at=1:7, labels=rasters,cex.axis=1.2,  cex.lab =2)

lines(grp$pl,col="red", type = "o",cex=1.5, lwd=2,pch = 1)
lines(grp$bm,col="blue", type = "o",cex=1.5, lwd=2,pch = 3)
lines(grp$xm,col="green", type = "o",cex=1.5, lwd=2,pch = 4)
lines(grp$cc,col="deepskyblue4", type = "o",cex=1.5, lwd=2,pch = 5)
```

SQL

These SQL Codes are used in IBD related mRNA analysis
Function for threshold calculation

```
ALTER FUNCTION [dbo].[findThreasold]
(
@degneg  as float,
@degdeg  as float,
@degibd  as float,
@ibdneg  as float,
@ibddeg  as float,
@ibdibd  as float
)
RETURNS @geneinteraction TABLE
(
totaldata bigint,
fgt  varchar(10),
sgt  varchar(10),
distn bigint
)
AS
BEGIN
    insert into @geneinteraction( totaldata ,        fgt  ,sgt  ,distn )

select COUNT(*)  totalint, FGT,SGT , (select count(distinct(FG)) from bintersum4
where FGT ='DEG' and SGT = 'NEG' and SCORE >= @degneg  )  dis
from bintersum4 where FGT ='DEG' and SGT = 'NEG' and SCORE >= @degneg group by FGT,SGT
union all

select COUNT(*) , FGT,SGT,(select count(distinct(FG)) from bintersum4
where FGT ='DEG' and SGT = 'DEG' and SCORE >= @degdeg ) dis
from bintersum4 where FGT ='DEG' and SGT = 'DEG' and SCORE >= @degdeg group by FGT,SGT
union all

select COUNT(*) , FGT,SGT,(select count(distinct(FG)) from bintersum4
where FGT ='DEG' and SGT = 'IBD' and SCORE >= @degibd ) dis
from bintersum4 where FGT ='DEG' and SGT = 'IBD' and SCORE >= @degibd group by FGT,SGT
union all

select COUNT(*) , FGT,SGT ,(select count(distinct(FG)) from bintersum4
where FGT ='IBD' and SGT = 'NEG' and SCORE >= @ibdneg ) dis
from bintersum4 where FGT ='IBD' and SGT = 'NEG' and SCORE >= @ibdneg group by FGT,SGT
union all


select COUNT(*) , FGT,SGT,(select count(distinct(FG)) from bintersum4
where FGT ='IBD' and SGT = 'DEG' and SCORE >= @ibddeg) dis
from bintersum4 where FGT ='IBD' and SGT = 'DEG' and SCORE >= @ibddeg group by FGT,SGT
union all

select COUNT(*) , FGT,SGT ,(select count(distinct(FG)) from bintersum4
where FGT ='IBD' and SGT = 'IBD' and SCORE >= @ibdibd) dis
from bintersum4 where FGT ='IBD' and SGT = 'IBD' and SCORE >= @ibdibd group by FGT,SGT

RETURN;
END;
```

Combine all web database gene

```
drop table genewebibd
select * into genewebibd from
(
select gene , 'IBD' gtype from (select * from gene ) t1 inner join
(
select * from
(
select t1.* , t1.DIS+t1.GWS+t1.HUG+t1.CTD total from
(
select * from
(
select gene ,'CTD' db from IBDCtdsc129
union all
select gene , 'HUG' db from IBDHugenet
union all
select gene , 'DIS' db from IBDDisGenet
union all
select gene , 'GWS' db from ibdgwas
) t1
PIVOT (count(db)
     FOR db in ([CTD],[HUG],[DIS],[GWS])) AS pvt
) t1
) t1 where t1.total >= 1
) t2 on ltrim(rtrim(t1.F1)) = ltrim(rtrim(t2.gene))
) t1

update gene  set GTYPE = 'IBD'  from gene inner join genewebibd on gene.F1 = genewebibd.gene
```

Gene labeling by IBD, DEG, NIBD and Formatting

```
drop table bintersum

select * into bintersum from
(
select ltrim(rtrim(t1.F3)) FG , t1.GTYPE FGT  ,t1.F6 SG, t2.GTYPE  SGT, t1.F7 SCORE from
(
select t1.F3 , t2.GTYPE,  t1.F6 , t1.F7  from binteraction t1 left join
gene t2 on ltrim(rtrim(t1.F3)) = t2.F1
) t1 left join gene t2 on t1.F6=t2.F1
) t1

drop table bintersum1
select * into bintersum1 from
(
select * from bintersum where FGT is not null
union all
select t1.FG , t2.GTYPE FGT , t1.SG , t1.SGT , t1.SCORE from
(
SELECT
    Split.a.value('.', 'VARCHAR(8000)') AS FG  , SG, SGT, SCORE
 FROM
 (
    SELECT FGT,SG,SGT , SCORE,
        CAST ('<M>' + REPLACE(FG, ',', '</M><M>') + '</M>' AS XML) AS Data
    FROM  (select replace(fg,'-',',') FG , FGT,SG,SGT , SCORE  from bintersum where FGT is null) t1
 ) AS A CROSS APPLY Data.nodes ('/M') AS Split(a)
) t1 left join gene t2 on ltrim(rtrim(t1.FG)) = t2.F1 where t2.F1 is not null
) t1
```

```
drop table bintersum2
select * into bintersum2 from
(
select * from
(
select * from bintersum1
except
select * from bintersum1 where (SG like '%-%' or SG like '%,%' ) and LEN(SG) > 5
) t1
union all
select t1.FG, t1.FGT , ltrim(rtrim(t1.SG)) , t2.GTYPE SGT , t1.SCORE from
(
SELECT  FG, FGT,
    Split.a.value('.', 'VARCHAR(8000)') AS SG , SCORE
 FROM
 (
    SELECT FGT,FG , SCORE,
       CAST ('<M>' + REPLACE(SG, ',', '</M><M>') + '</M>' AS XML) AS Data
    FROM  (select replace(SG,'-',',') SG , FGT,FG , SCORE  from bintersum1 where (SG like '%-%' or SG like '%,%' ) and LEN(SG)
> 5) t1
 ) AS A CROSS APPLY Data.nodes ('/M') AS Split(a)
) t1 left join gene t2 on ltrim(rtrim(t1.SG)) = t2.F1

)  t1

select * from bintersum2 where sGT is null

update bintersum2 set FG= LTRIM(rtrim(fg)) , SG = ltrim(RTRIM(sg))

drop table bintersum3
select * into bintersum3
from
(
select * from
(
select
ROW_NUMBER() OVER(PARTITION BY fg , sg  ORDER BY score DESC) AS sl  , FG , FGT , SG , SGT , SCORE
from bintersum2
) t1  where t1.sl = 1
) t1

select * from (select * from bintersum2 where SGT is null) t1 left join cgene866 t2 on t1.SG = ltrim(rtrim(t2.ibd866))

select * from bintersum3 t1 left join cgene866 t2 on t1.SG = ltrim(rtrim(t2.ibd866))
where t2.ibd866 is not null


select * into bintersum31 from
(
select * from bintersum3 where SGT is not null
union all
select sl ,FG, FGT , SG , 'IBD' sgt , score from (select * from bintersum3 where SGT is null) t1 left join cgene866 t2 on t1.SG =
ltrim(rtrim(t2.ibd866))
where t2.ibd866 is not null
union all
select sl ,FG, FGT , SG , 'NEG' sgt , score from (select * from bintersum3 where SGT is null) t1 left join cgene866 t2 on t1.SG
= ltrim(rtrim(t2.ibd866))
where t2.ibd866 is  null
) t1
```

```
------------------------ for only 318 gene

drop table bintersum31
select * into bintersum31 from
(
select * from bintersum3 where SGT is not null
union all
select sl ,FG, FGT , SG , t2.GTYPE sgt , score from (select * from bintersum3 where SGT is null) t1 left join gene t2 on t1.SG =
ltrim(rtrim(t2.F1))
where t2.gtype is not null
union all
select sl ,FG, FGT , SG , 'NEG' sgt , score from (select * from bintersum3 where SGT is null) t1 left join gene t2 on t1.SG =
ltrim(rtrim(t2.F1))
where t2.gtype is  null
) t1

select * from bintersum31 where SGT is null

drop table bintersum4
select * into bintersum4 from
(
select * from bintersum31
except
select * from bintersum31 where SG= fg
) t1

select * from bintersum4 where sg is null
delete from bintersum4 where sg is null
```

Selecting interaction set by thresholds

```
select * from IBDG.dbo.findThreasold (0.85, 0.1, 0.1, 0.72, 0.1, 0.1)

select COUNT(distinct(fg)) , fgt from IBDG.dbo.finddata(0.85, 0.1, 0.1, 0.72, 0.1, 0.1)
group by  fgt

select COUNT(distinct(sg)) , sgt from IBDG.dbo.finddata(0.85, 0.1, 0.1, 0.72, 0.1, 0.1)
group by  sgt

select * from IBDG.dbo.finddata(0.85, 0.1, 0.1, 0.72, 0.1, 0.1)

select * from cluster
```

Convert comma separated value of a cluster to rows

```
drop table clusterh
select cn , LTRIM(rtrim(cg)) cg into clusterh from
(
SELECT cn,
    Split.a.value('.', 'VARCHAR(8000)') AS cg
 FROM
(
  select cn , CAST ('<M>' + REPLACE(clds, ',', '</M><M>') + '</M>' AS XML) AS Data    from cluster
) AS A CROSS APPLY Data.nodes ('/M') AS Split(a)
) t1


select * into genetotal from
(
select distinct fg , fgt from
(
select FG , FGT  from bintersum4
union all
select SG , SGT  from bintersum4
) t1 group by FG , FGT
) t1
```

Formatting the data for fisher exact test

```
select t1.* , t1.ibd + t1.nibd +t1.ogn+t1.nogn from
(
select t1.cn , t1.clds , t2.ibd + t2.deg + t2.neg totalgn , IBD ibd, 291 - IBD nibd, t2.DEG + t2.NEG ogn ,
4765 - t2.DEG - t2.NEG  nogn    from cluster t1 left join
(
select * from
(
select cn , fgt from clusterh t1 left join genetotal t2 on t1.cg = t2.FG
) t1
PIVOT (count(fgt) FOR fgt IN (IBD,DEG,NEG)) AS pvt
) t2 on t1.cn = t2.cn
) t1 order by t1.ibd desc
```

```
2072        DEG
291         IBD
2693        NEG
```

Select Data Using threshold

```
select COUNT(*) , fgt from
(
select distinct fg , fgt from
(
select fg , fgt  from IBD.dbo.finddata(0.85, 0.1, 0.1, 0.72, 0.1, 0.1)
union all
select sg , sgt  from IBD.dbo.finddata(0.85, 0.1, 0.1, 0.72, 0.1, 0.1)
) t1 group by fg , fgt
) t1 group by fgt

select * from clusterp
-----------------------------------
drop table clusterph

select * into clusterph from
(
SELECT cn,
    Split.a.value('.', 'VARCHAR(8000)') AS cg
 FROM
(
  select cn , CAST ('<M>' + REPLACE(clds, ',', '</M><M>') + '</M>' AS XML) AS Data    from clusterp
) AS A CROSS APPLY Data.nodes ('/M') AS Split(a)
) t1
select * from clusterph
update clusterph set cg = LTRIM(rtrim(cg))
--------------
```

Separation of different type of genes from each cluster

```
drop table clusterptype
select * into clusterptype from
(
Select distinct ST2.cn,
   substring(
      (
         Select ','+ST1.cg  AS [text()]
         From (
                        select cn, cg  from clusterph t1 left join

                                                            (
                                                            select distinct fg , fgt from
                                                            (
                                                            select fg , fgt from bintersum4
                                                            union all
                                                            select sg , sgt from bintersum4
                                                            ) t1 group by fg , fgt
                                                            ) t2 on t1.cg = t2.FG  where FGT = 'IBD'
                                                            )  ST1
         Where ST1.cn = ST2.cn
         ORDER BY ST1.cn
         For XML PATH ('')
      ), 2, 1000) Genes ,'IBD' type
From
(
select cn, cg  from clusterph t1 left join

(
select distinct fg , fgt from
(
select fg , fgt from bintersum4
union all
select sg , sgt from bintersum4
) t1 group by fg , fgt
) t2 on t1.cg = t2.FG  where FGT = 'IBD'
) ST2
union all

Select distinct ST2.cn,
   substring(
      (
         Select ','+ST1.cg  AS [text()]
         From (
                                                            select cn, cg  from clusterph t1 left join

                                                            (
                                                            select distinct fg , fgt from
                                                            (
                                                            select fg , fgt from bintersum4
                                                            union all
                                                            select sg , sgt from bintersum4
                                                            ) t1 group by fg , fgt
                                                            ) t2 on t1.cg = t2.FG  where FGT = 'DEG'
                                                            )  ST1
         Where ST1.cn = ST2.cn
         ORDER BY ST1.cn
         For XML PATH ('')
      ), 2, 1000) Genes ,'DEG' type
From
(
select cn, cg  from clusterph t1 left join

(
select distinct fg , fgt from
(
select fg , fgt from bintersum4
union all
select sg , sgt from bintersum4
) t1 group by fg , fgt
) t2 on t1.cg = t2.FG  where FGT = 'DEG'
) ST2

union all

Select distinct ST2.cn,
   substring(
      (
         Select ','+ST1.cg  AS [text()]
         From (
                                                    select cn, cg  from clusterph t1 left join

                                                    (
                                                    select distinct fg , fgt from
                                                    (
                                                    select fg , fgt from bintersum4
                                                    union all
                                                    select sg , sgt from bintersum4
                                                    ) t1 group by fg , fgt
                                                    ) t2 on t1.cg = t2.FG  where FGT = 'NEG'
                                                    )  ST1
         Where ST1.cn = ST2.cn
         ORDER BY ST1.cn
         For XML PATH ('')
      ), 2, 1000) Genes ,'NEG' type
From
(
select cn, cg  from clusterph t1 left join

(
select distinct fg , fgt from
(
select fg , fgt from bintersum4
union all
select sg , sgt from bintersum4
) t1 group by fg , fgt
) t2 on t1.cg = t2.FG  where FGT = 'NEG'
) ST2
) t1
```

Comparision of the discovered IBD genes to different database with p_value, (descending order) and cluster list attached to them

```
select t1.* , IBDDisGenet+IBDCtdsc129+IBDHugenet+IBDgwas fourc , t2.assigncluster, t2.frq , LEN(assigncluster)
cllen from
(
select * from
(
select   ROW_NUMBER() OVER(PARTITION BY cg ORDER BY pval asc) sl ,
 t1.* from
(
select t1.* , case when t2.sc is null then 0 else 1 end IBDgwas from
(
select t1.* , case when t2.sc is null then 0 else 1 end IBDHugenet from
(
select t1.* , case when t2.sc is null then 0 else 1 end IBDCtdsc129 from
(
select t1.* , case when t2.sc is null then 0 else 1 end IBDDisGenet
from
(
select t1.* , t2.pval from
(
select t1.cn , t1.cg , t2.FGT from  clusterph  t1 left join
(
select distinct fg , fgt from
(
select fg , fgt from bintersum4
union all
select sg , sgt from bintersum4
) t1 group by fg , fgt
) t2 on t1.cg = t2.FG
) t1 left join clusterp t2 on t1.cn = t2.cn
) t1 left join IBDDisGenet  t2 on t1.cg = t2.gene
) t1 left join IBDCtdsc129  t2 on t1.cg = t2.gene
) t1 left join IBDHugenet t2 on t1.cg = t2.gene
) t1 left join ibdgwas t2 on t1.cg = t2.gene
) t1
) t1
) t1 left join
(
select t1.cg , ltrim(rtrim(t1.assigncluster)) assigncluster , t2.frq from
(
SELECT  cg
      ,STUFF((SELECT ', ' + CAST(cn AS VARCHAR(10)) [text()]
        FROM clusterph
        WHERE cg = t.cg
        FOR XML PATH(''), TYPE)
        .value('.','NVARCHAR(MAX)'),1,2,' ') assigncluster
FROM clusterph t
GROUP BY cg
) t1 left join (select cg, COUNT(*) frq from clusterph group by cg) t2 on t1.cg = t2.cg
) t2 on t1.cg = t2.cg
WHERE cn IN (15,28,54,102,133,169) and FGT <> 'IBD'
```