Manual for BCT (Version 1.1)

Overview

1. What is BCT?

BCT is Matlab toolbox which is designed to compare between biclustering algorithms.
You can compare between these algorithms based on many points:

1. The percentage of enriched biclusters for each algorithm.
2. The ability of the algorithms to recover selected patterns.

BCT block diagram is shown in Figure 1. First, as illustrated in this figure BCT input are the biclustering output files, which contains the biclusters results from biclustering algorithms which are implemented in BCT or implemented in one of available biclustering toolbox like BicAT toolbox ,Bivisu program , ……
Second, function enrichment was analyzed for each biclusters using GeneMerge Perl program by setting sufficient significance level and interested GO category. Third, as the number of generated biclusters varies strongly among the considered methods, a postprocessing filtration procedure has been applied to the output of the algorithms to provide a common basis for the comparison. Finally, using one of comparison methodology which was implemented in BCT, the user could test the performance of various algorithms.

In appendix A we explain in more details the biclustering algorithms which are implemented in BCT.

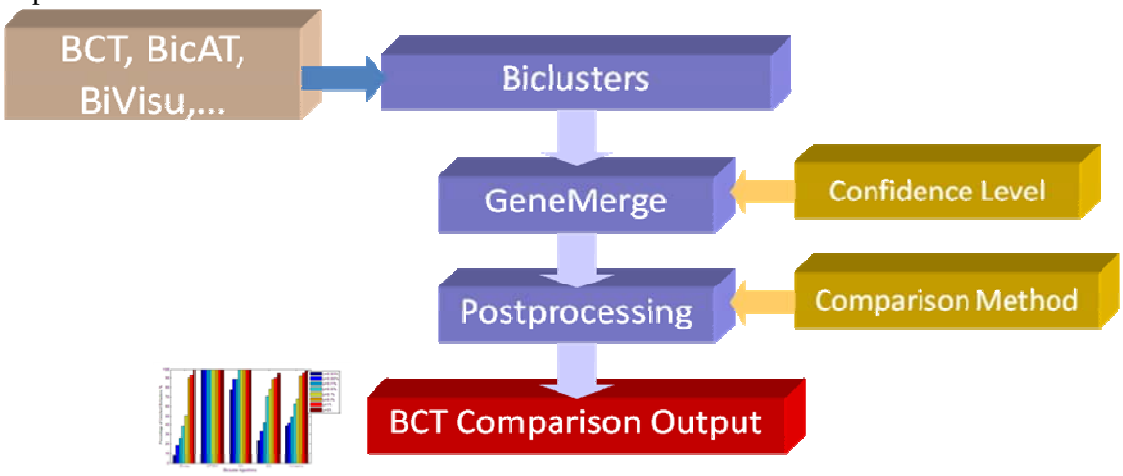


Fig. 1. Blook diagram of the BCT.

2. System requirement

We test BCT on a desktop PC with P4 1.8G CPU and 2.0 G memory running windows 7 operating system and Matlab 7.11.0.

3. Startup

1. Download BCT.zip from ……
2. Unzip the file into a directory, e.g. C:\.
A folder named 'BCT' should be created which contains BCT functions and all association files as shown in Fig 2.
3. Execute Matlab, and change the current directory to the directory to the files folder (i.e C:\BCT\files).
4. Type the following in the command window and press enter
guide
5. In the pop-up dialog, select "Open Existing GUI" and then press "browse" button. A separate dialog window appears.
6. Locate the file "BCT.fig" and press "open" button. After that, the GUI of BCT is open with a layout editor as shown in Fig.3.

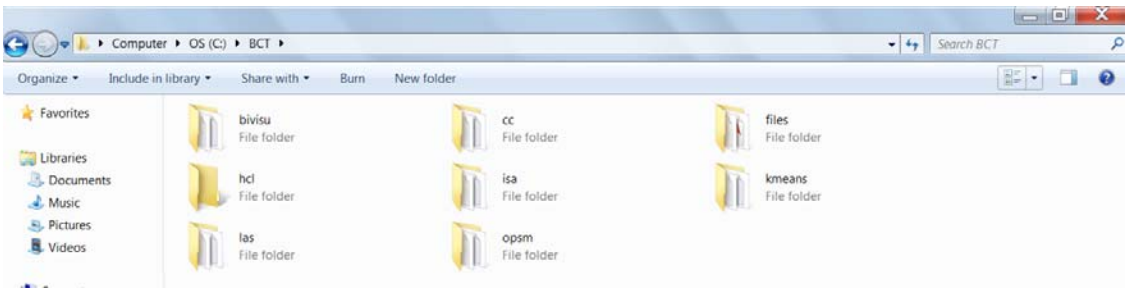


Fig. 2. Main folders of BCT

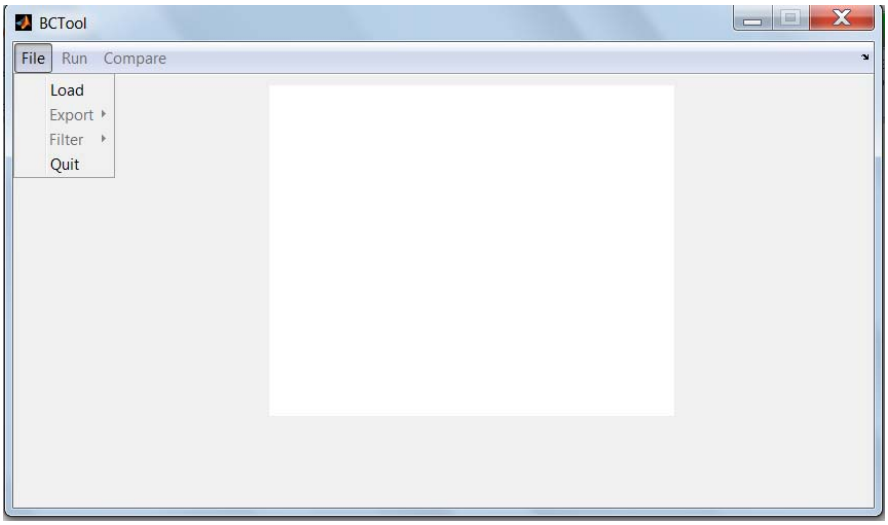
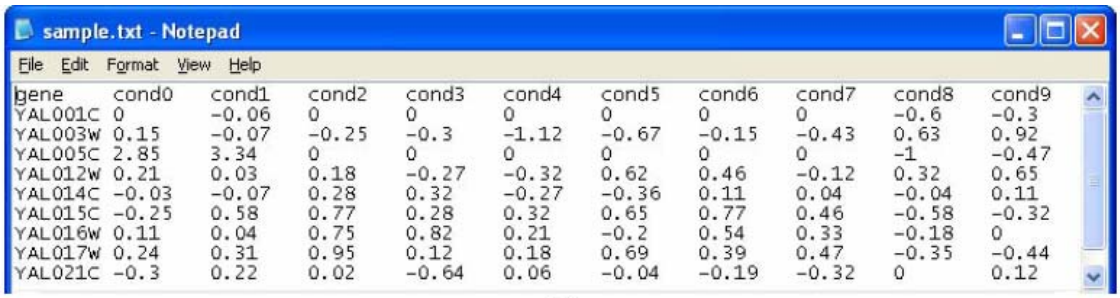


Fig. 3. Main window of BCT

4. Loading Data

BCT accepts gene expression matrix stored in text files in formats as shown in Fig. 4. In this format, row and column headers (such as gene names and experimental conditions respectively) are included. Gene names here must be as ORF type.

To load a data file,
1) Choose “Load” in the “File” menu.
2) In the pop-up dialog as shown in Fig. 5, browse for the data file or type the path and filename of the data file in the textbox of “File name”. After pressing the “Open” button, the selected file is loaded onto BCT.



gene	cond0	cond1	cond2	cond3	cond4	cond5	cond6	cond7	cond8	cond9
YAL001C	0	-0.06	0	0	0	0	0	0	-0.6	-0.3
YAL003W	0.15	-0.07	-0.25	-0.3	-1.12	-0.67	-0.15	-0.43	0.63	0.92
YAL005C	2.85	3.34	0	0	0	0	0	0	-1	-0.47
YAL012W	0.21	0.03	0.18	-0.27	-0.32	0.62	0.46	-0.12	0.32	0.65
YAL014C	-0.03	-0.07	0.28	0.32	-0.27	-0.36	0.11	0.04	-0.04	0.11
YAL015C	-0.25	0.58	0.77	0.28	0.32	0.65	0.77	0.46	-0.58	-0.32
YAL016W	0.11	0.04	0.75	0.82	0.21	-0.2	0.54	0.33	-0.18	0
YAL017W	0.24	0.31	0.95	0.12	0.18	0.69	0.39	0.47	-0.35	-0.44
YAL021C	-0.3	0.22	0.02	-0.64	0.06	-0.04	-0.19	-0.32	0	0.12

Fig. 4. Example of BCT input file.

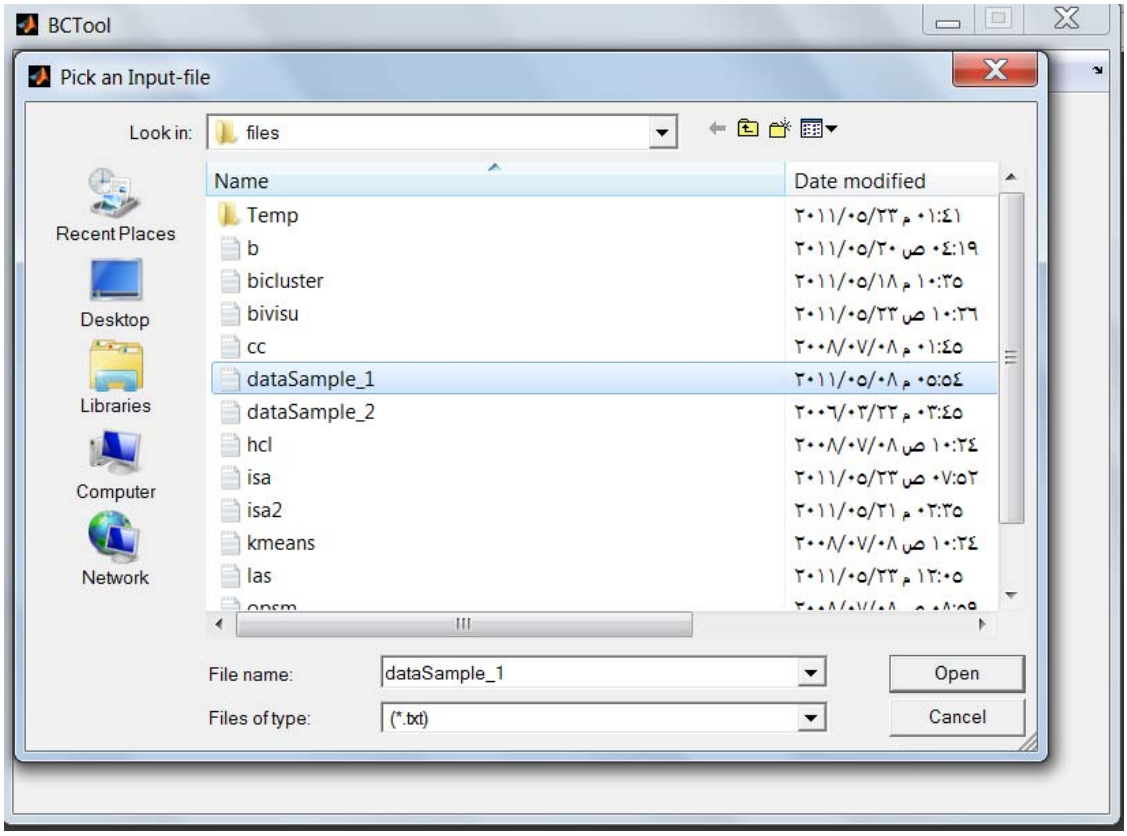


Fig.5 A pop-up dialog which prompts for an input file.

5. Biclusters Detection

Three biclusterig algorithms were implemented in BCT which are: ISA, LAS, and Bivisu. The details of these algorithms are explained in appendix A.

After entering appropriate values of parameters, click “OK”. Bicluster detection is then performed. The processing time depends on the data size and parameters setting. The detected biclusters will be arranged in the descending order of their sizes.

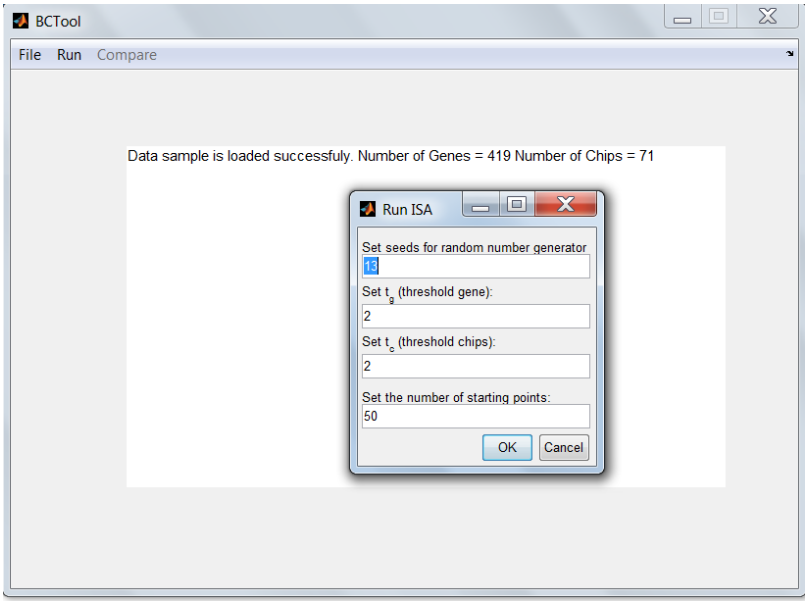


Fig. 6. ISA Main windows.

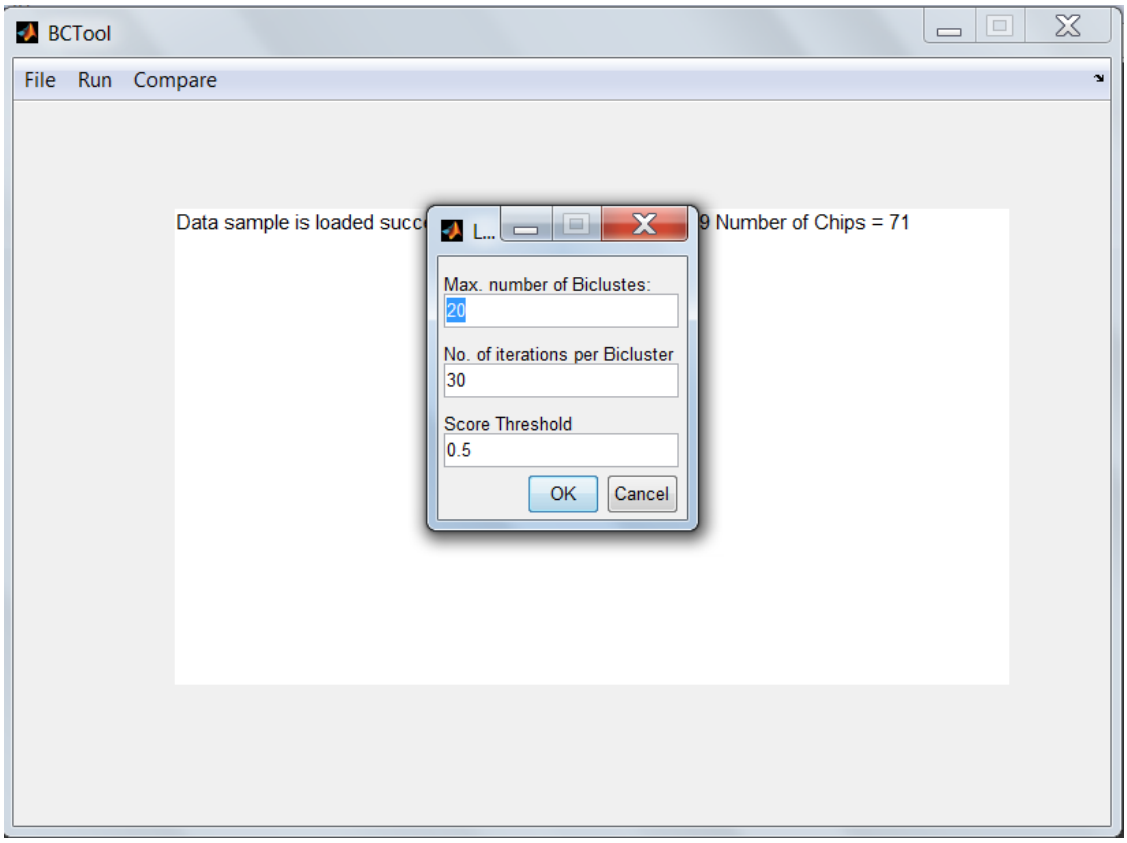


Fig. 7. LAS Main windows.

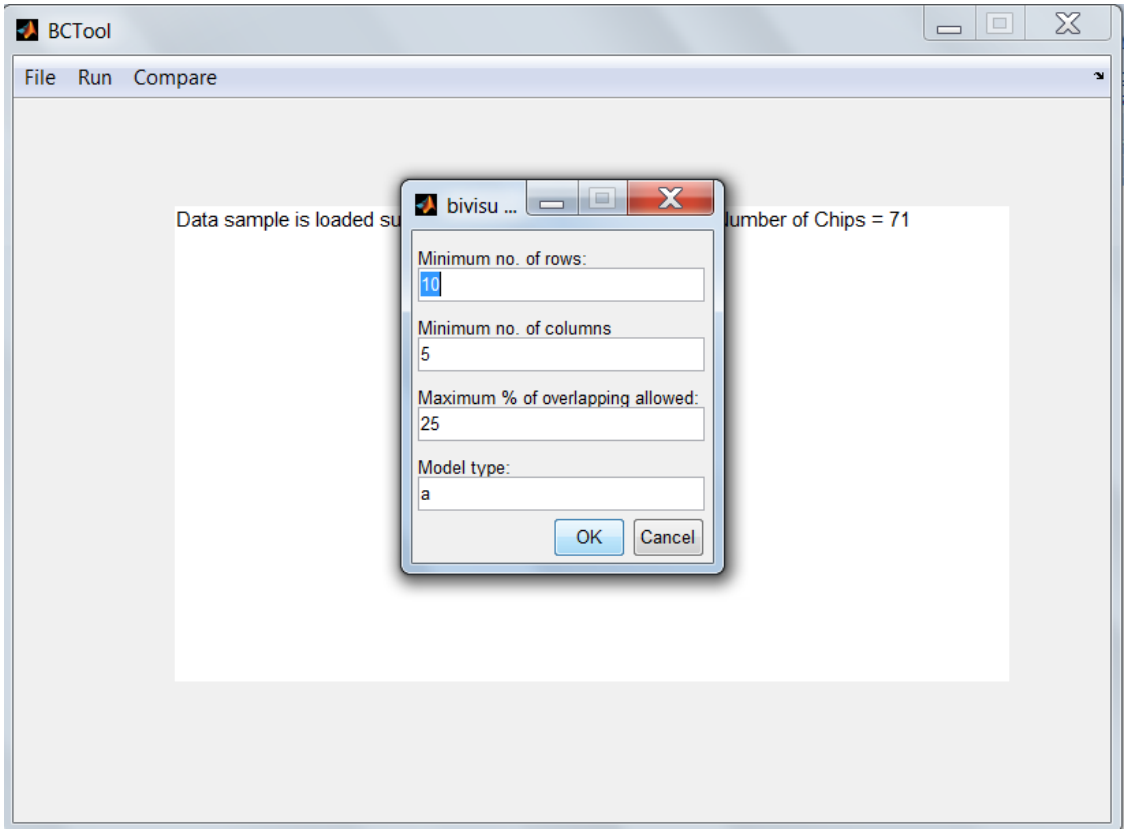


Fig. 8. Bivisu Main windows.

6. Biclusters Filtration

Filtering can refine the biclustering result without reperforming the expensive biclustering process. It can be set by choosing “Filter” in “Tools” menu as shown in Fig.9. Four criteria can
be specified:

• *Minimum no. of rows*: it specifies the minimum number of rows in a bicluster to be displayed.

• *Minimum no. of columns*: it specifies the minimum number of columns in a bicluster to be displayed.

• *Maximum no. of biclusters*: it specifies the maximum number of biclusters retained in the biclustering results.

• *Maximum % of overlap allowed*: The maximum percentage of overlap allowed either in rows or columns (depending on which dimension has smaller overlap) between two biclusters.

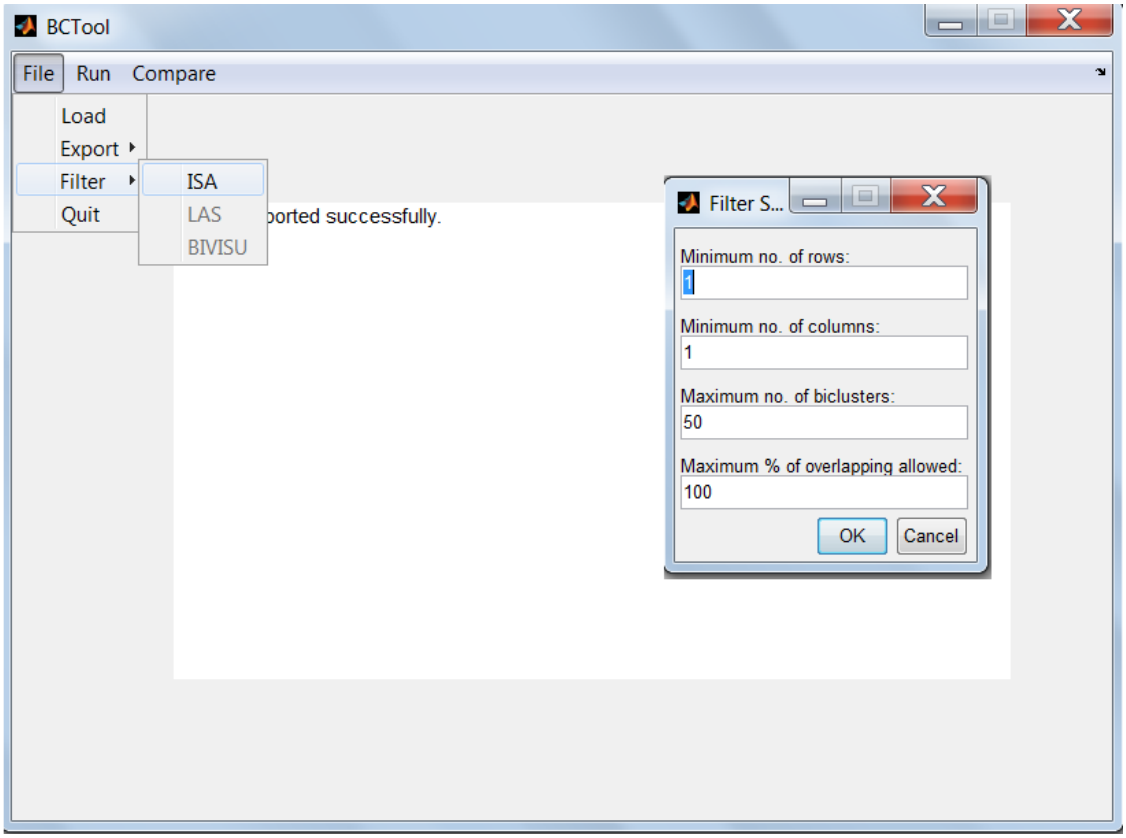


Fig. 9. A dialog box for entering filtering-related parameters.

7. Saving Output

Biclustering results can be exported to text files by choosing “File” -> “Export”. A popup window as shown in Fig. 10. is then displayed.

Note:
Biclustering results must be saved as algorithms title and in the directory of folder files. For example, ISA results must be saved as ISA.txt in c:\BCT\files\.

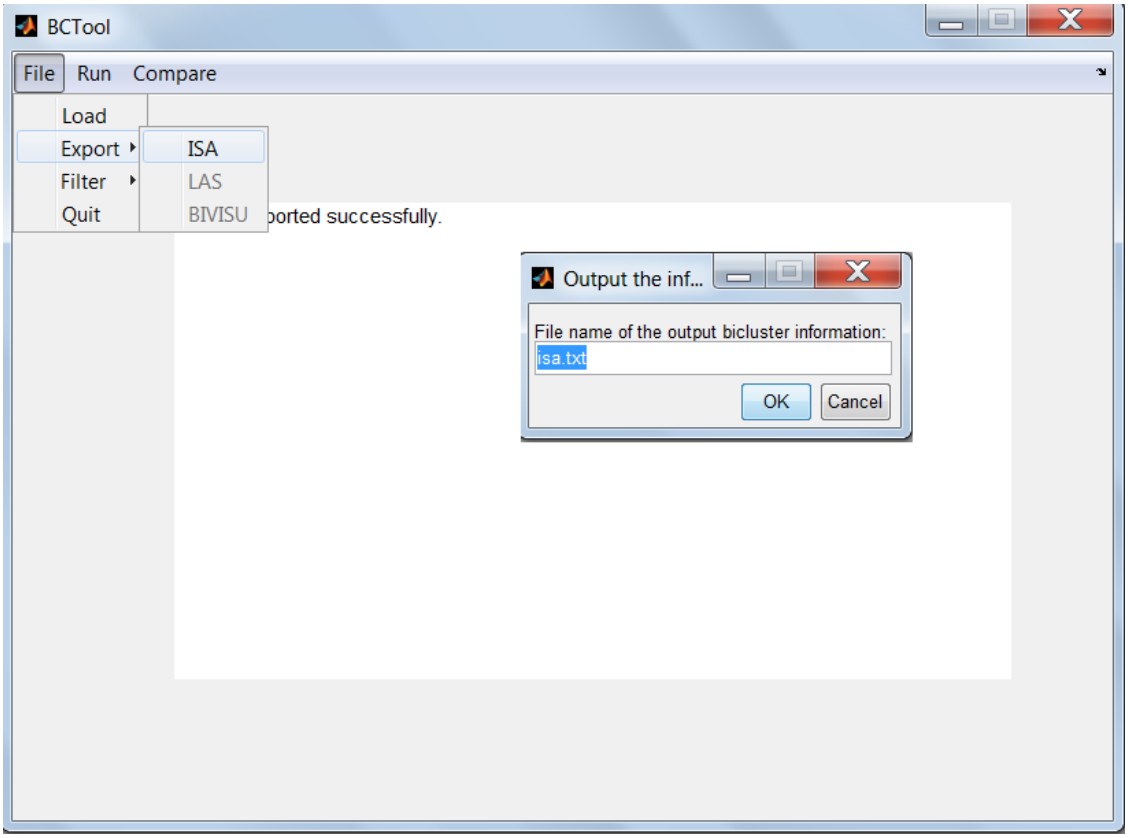


Fig. 10. A dialog box for saving biclustering results.

Examples of output files are shown in Fig.11. The output file for the biclustering results gives information of each bicluster. The first line gives the size of the bicluster (# of genes and conditions).
The second and third lines show the genes and conditions names.

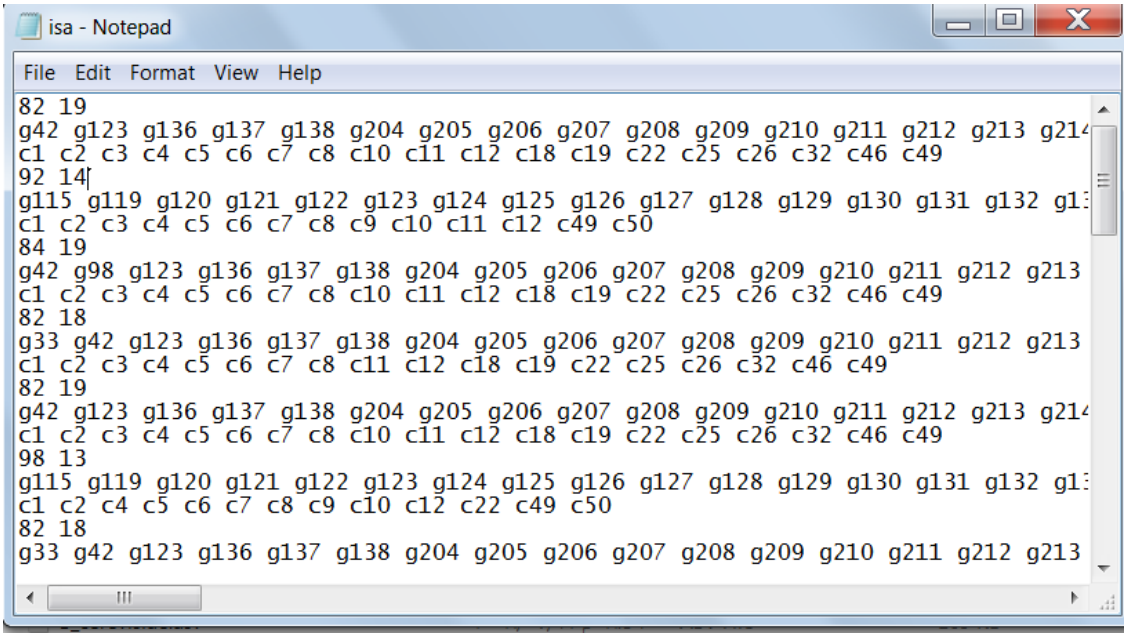


Fig. 11. Examples of output files.

8. Biclusters Comparison

We have to define many important terms for comparing biclusters:
1. The percentage of enriched or overrepresented biclusters: This percentage is calculated for each algorithm with one or more GO term per multiple significance levels (p-values) for each algorithm using the below equation:

$$Percentage\ of\ Enriched\ Biclusters = \frac{Number\ of\ Enriched\ Biclusters}{Total\ Number\ of\ Biclusters} \times 100$$

2. Percentage of annotated genes per each bicluster: Sometimes even the bicluster is enriched, it contains few annotated genes. So we defined the percentage of annotated genes per each bicluster as more specific comparison metric as following:

$$Percentage\ of\ Annotated\ Genes\ per\ Each\ Bicluster = \frac{No\ of\ Genes\ Sharing\ GO\text{–}Term\ in\ aBicluster}{Total\ Number\ of\ Genes\ in\ this\ Bicluster}$$

BCT provides three reasonable methods for comparing the results of different biclustering algorithms by: (please if you are not familiar by Gene ontology and hypergometric test go to appendix B)

1. **Option 1**: Identifying the percentage of enriched or overrepresented biclusters with one or more GO term per multiple significance levels for each algorithm. A bicluster is said to be significantly overrepresented (enriched) with a functional category if the p-value of this functional category is lower than the preset threshold P-value.

The results are displayed using a histogram for the entire compared algorithms at the different preset significance levels, and the algorithm which gives higher proportion of enriched biclusters per all significance levels is considered to be the optimum one as it does group effectively the genes sharing similar functions in the same bicluster.

2. **Option 2:** Identifying the percentage of annotated genes per each enriched bicluster.

3. **Option 3:** Estimating the algorithms predictability power to recover interesting pattern.

Genes whose transcription is responsive to a variety of stresses have been implicated in a general yeast response to stress. Other gene expression responses appear to be specific to particular environmental conditions. BCT Compare biclustering methods based on which of them could recover known patterns in experimental datasets. For example, in Gasch measure changes in transcript levels over time responding to panel of environmental changes. So it was expected to find enriched biclusters with one of response to stress (GO:0006950) Gene Ontology category like response to heat (GO:0009408), response to cold (GO:0009409) and response to glucose starvation(GO:0042149).

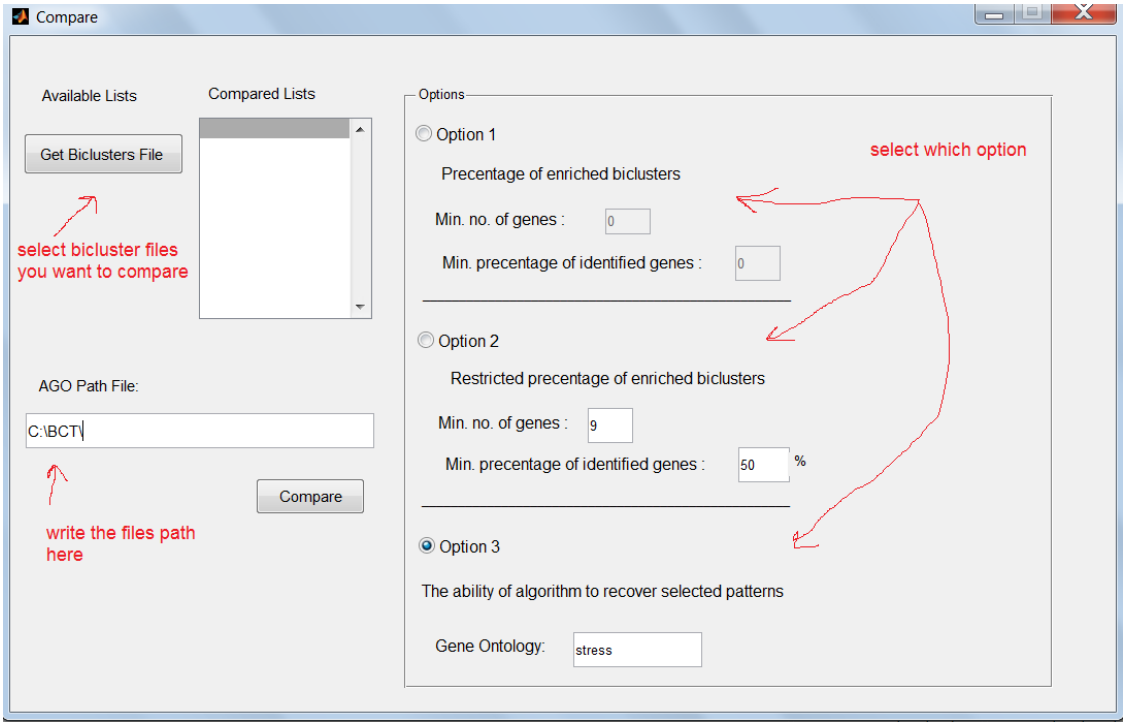


Fig. 12. A dialog box for comparing biclustering results.

9. Appendix A: Biclustering Algorithms

9.1 Clustering vs. Biclustering

Detecting groups (clusters) of closely related objects is an important problem in bioinformatics and data mining in general. Laboratories apply every existing clustering method to their microarray data sets, hoping to find some significant genes or clusters. In this section we will first give basic background on clustering and biclustering. We then describe the major differences between them.

What is Clustering?

A large number of clustering definitions can be found in the literature. The simplest definition is shared among all and includes one fundamental concept: the grouping together of similar data items into clusters.
Clustering is an important explorative statistical analysis of gene expression data. It aims to identify and group genes that exhibit similar expression patterns over several conditions and also group the conditions based on the expression profiles across set of genes. The successful clustering approach should guarantee two criteria which are homogeneity high similarity between elements in the same cluster, and separation – low similarity between elements from different clusters. When homogeneity and separation are precisely defined, those are two opposing objectives: The better the homogeneity the poorer the separation, and vice versa. Several algorithmic techniques were previously used for clustering gene expression data, including hierarchical clustering, self organizing maps, and graph theoretic approaches.

K-means:

K-means is a classical clustering algorithm invented in 1956 to classify or to group objects (genes) based on attributes or features (experimental conditions) into K number of groups (clusters). K is positive integer number and assumed to be known. Kmeans computational approach starts by placing K points into the space represented by the objects that are being clustered. These points represent initial group centroids. We can take any random objects as the initial centroids or the first K objects in sequence can also be used as the initial centroids. Then the K means algorithm will do the four steps below until convergence:
1. Determine the centroids coordinate.
2. Determine the distance of each object to the centroids using the Euclidean distance which is defined as:

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Where p is the object (gene expression) value of i condition, q is centroid point value of i condition and n is the total number of conditions.
3. Group the objects based on minimum distance.
4. Iterate the above steps till no object moves its assigned group.

Each iteration of k-means modifies the current partition by checking all possible modifications of the solution, in which one element is moved to another cluster. This is done by reducing the sum of distances between objects and the centers of their clusters.
This procedure is repeated until no further improvement is achieved (No object move the group) and all the objects are grouped into the final required number of clusters.
A disadvantage of K-means algorithm could be perceived in the need to specify the number of clusters K as a parameter value prior to running the algorithm. In cases where there is no expectation about K, user has to make trails with several values of K or use external techniques to guess the no of clusters may be exist.

Hierarchical clustering (HCL):

Hierarchical clustering does not partition the genes into subsets. Instead it builds a down-top hierarchy of clusters using agglomerative methods or top - down hierarchy of clusters using divisive methods. The traditional graphical representation of this hierarchy is called dendrogram tree. The divisive method begins at the root and starts to breaks up clusters whose having low similarity. Whereas, the Agglomerative method begins at the leaves of the tree and starts with an initial partition into single element clusters and successively merges clusters until all elements belong to the same cluster. (See Figure A-1) The agglomerative method is widely used than the divisive one which is not generally available, and rarely has been applied. The idea of the agglomerative method can be summarized as following: Given a set of N items (genes in our case) to be clustered, and an N*N distance (or similarity) matrix,
1. Assign each item to a cluster, so you have N clusters, each containing just one item.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster.
3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.
In Step 3, distance or similarity measurements between the merged clusters and all the other clusters can be calculated in one of three schemes: single-linkage, complete linkage and average-linkage.

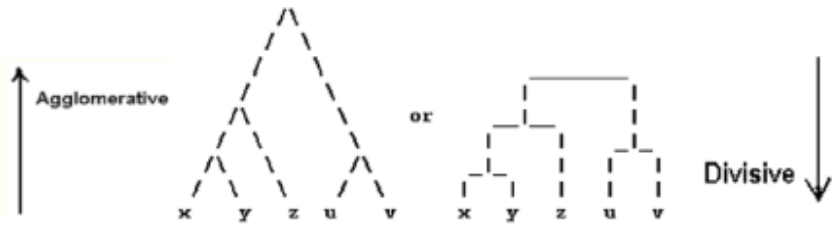


Figure A-1: HCL:Agglomerative and Divisive Methods.

What is Biclustering?

Traditional clustering approaches such as k-means and hierarchical clustering put each gene in exactly one cluster based on the assumption that all genes behave similarly in all

conditions. However, recent understanding of cellular processes shows that it is possible for subset of genes to be co expressed under certain experimental conditions, and at the same time; to behave almost independently under other conditions. From this context, a new two mode clustering approach called biclustering or co-clustering has been introduced to group the genes and conditions in both dimensions simultaneously.
This allows finding subgroups of genes that show the same response under a subset of conditions, not all conditions. Also, genes may participate in more than one function, resulting in one regulation pattern in one context and a different pattern in another.
Example, if a cellular process is only active under specific conditions and there is a gene participates in multiple pathways that are differentially regulated, one would expect this gene to be included in more than one cluster; and this cannot be achieved by traditional clustering techniques.

Iterative Signature Algorithm (ISA)

The ISA algorithm is a novel method for the biclustering analysis of large-scale expression data. It is an efficient algorithm based on the iterative application of the signature algorithm. ISA considers a bicluster to be a transcription module which can be defined as a set of coexpressed genes together with the associated set of regulating conditions (Figure A-2). Starting with an initial set of genes, all samples (conditions) are scored with respect to this gene set and those samples are chosen for which the score exceeds a certain threshold (usually defined by the user). In the same way, all genes are scored regarding the selected samples and a new set of genes is selected based on another user-defined threshold. The entire procedure is repeated until the set of genes and the set of samples converge and do not change anymore.
Multiple biclusters can be discovered by running the ISA algorithm on several initial gene sets. This approach requires identification of a reference gene set which needs to be carefully selected for good quality results. In the absence of pre-specified reference gene set, random set of genes is selected at the cost of results quality.

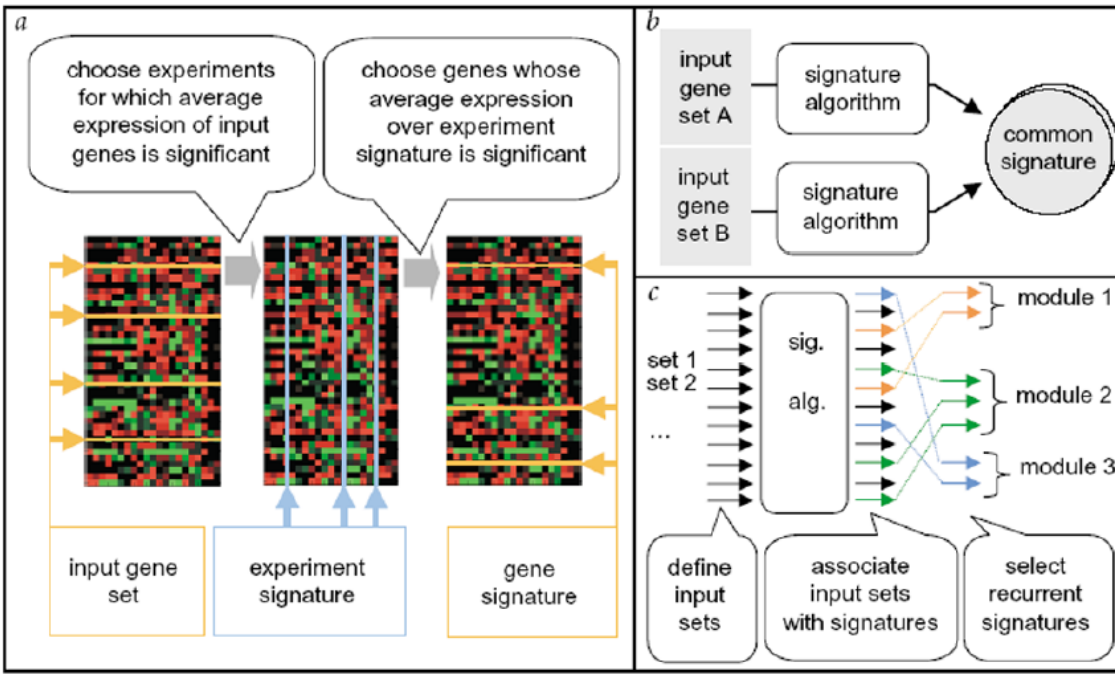


Figure A-2: The recurrence signature method. a, The signature algorithm. b, Recurrence as a reliability measure. The signature algorithm is applied to distinct input sets containing different subsets of the postulated transcription module. If the different input sets give rise to the same module, it is considered reliable. c, General application of the recurrent signature method.

LAS:

The procedure, which is called LAS (for Large Average Submatrix), finds large average submatrices within a given real-valued data matrix operates in an iterative fashion, and is based on a simple significance score that trades off between the size of a submatrix and its average value.

A connection is established between maximization of the significance score and the minimum description length principle. Using a simple Gaussian null model for the observed data, we assign a significance score to each submatrix U of the data matrix using a Bonferroni-corrected p-value that is based on the size and average value of the entries of U. The Bonferroni correction accounts for multiple comparisons that arise when searching among many submatrices for a submatrix having a large average value. In addition, the correction acts as a penalty that controls the size of discovered submatrices. The LAS score function is based on the normal CDF, and is sensitive to departures from normality that arises from heavy tails in the empirical distribution of the expression values. Outliers can give rise to submatrices that, while highly significant, have very few samples or variables.

BIVISU:

BiVisu is an open-source software tool for detecting and visualizing biclusters embedded in a gene expression matrix. BiVisu has been developed in Matlab and is available at http://www.eie.polyu.edu.hk/~nflaw/Biclustering/.

In BiVisu, row clustering is first performed by comparing every two columns and potential row clusters in each column pair are identified. These row clusters are then intersected to identify column pairs that can be merged together to form big biclusters. Note that the intersection process would not be performed if there is a significant drop in the number of rows after merging certain columns onto the current biclusters. The type of biclusters found is determined by how columns are compared. If the column pair is compared by calculating their differences in expression levels, additive-related biclusters are found. If the column pair is compared by calculating the ratio of their expression levels, multiplicative-related biclusters are found.

Appendix B:

Gene Ontology

In the last five years, biologists faced a problem of annotating the completed genome sequences especially for the Drosophila and the S.cerevisiae species and the organizations of the complex databases start to provide their own classification terminologies.

Consequently, these wide variations in terminologies and annotations inhibit effective searching by both computers and people. For example, if biologist was searching for new targets for antibiotics, If one database describes these molecules as being involved in 'translation', whereas another uses the phrase 'protein synthesis', it will be difficult for biologist and even harder for a computer to find functionally equivalent terms.

Therefore formal and explicit specifications of the gene annotation terms (in the shape of well-structured and controlled vocabularies) used and the relationships between them have been defined. This is called Gene Ontology and referred as GO. Using GO, biologists and researchers have systematic consistent classification of genes functions, in the form of a dictionary of functional terms that are hierarchically structured to allow both attribution and querying at different levels of granularity (See Figure B-1).

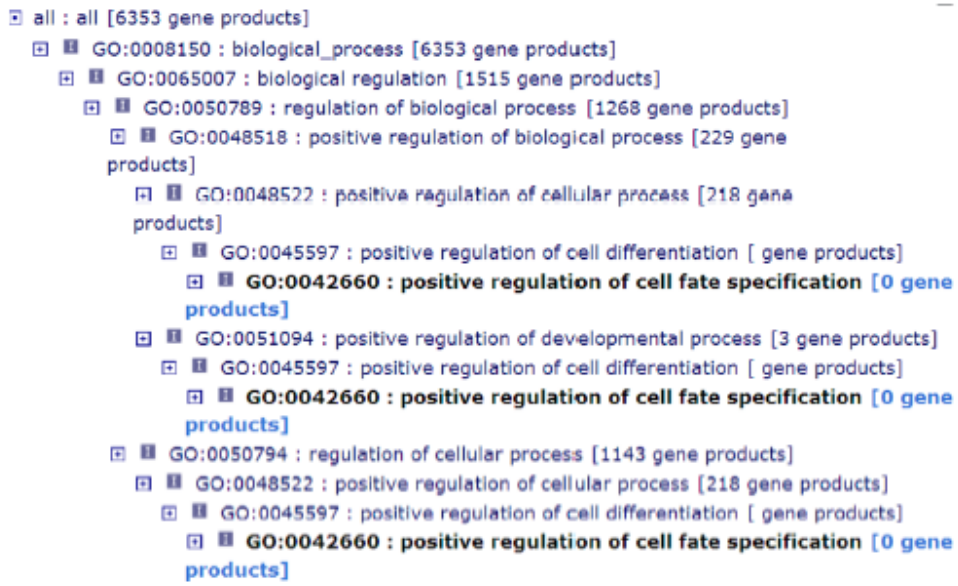


Figure B-1: Tree view of Biological Process Gene Ontology Category of S.cerevisiae.

The building blocks of the Gene Ontology are the terms (sometimes called functional classes or functional categories). Each GO term has a unique number and a textual name. E x, GO: 0042660: positive regulation of cell fate specification. Each GO term is assigned to one of the three subontologies(Figure B-2) in GO: biological process, molecular function and cellular component.

1. Biological process (GO:0008150): A function represented in a series of events and activities of a living system, mediated by protein or RNA.
2. Molecular function (GO:0003674): A function associated with the biochemical activity (including specific binding to ligands or structures) of a gene product.

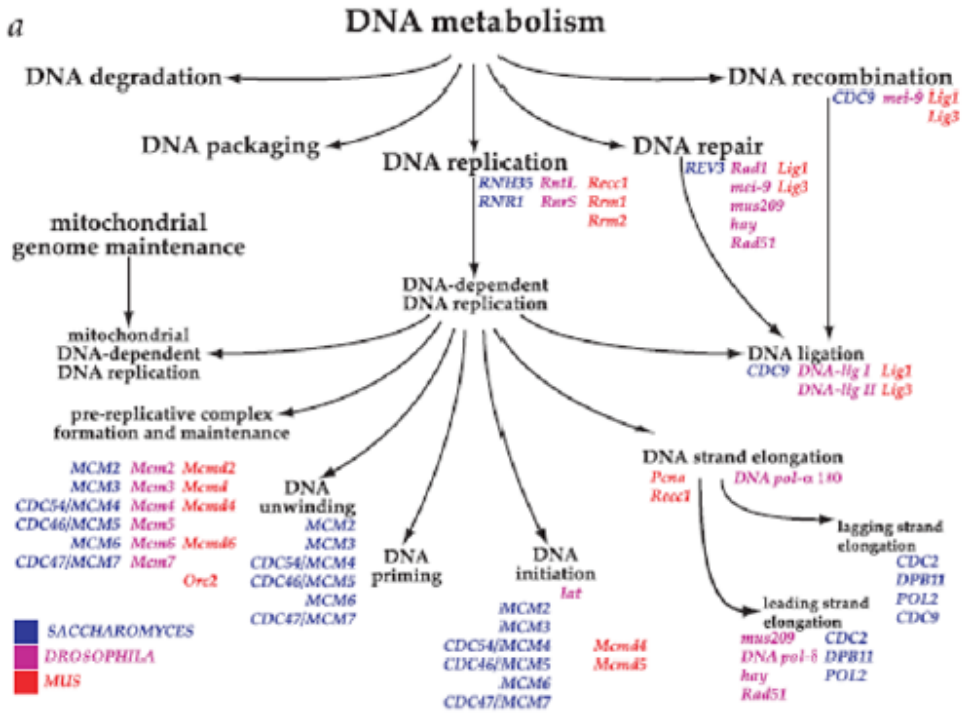


Figure B-2: Example of Gene Ontology to Illustrate the Structure and Style used by GO to Represent the Gene Ontologies and to Associate Genes with Nodes within an Ontology.

3. Cellular component (GO:0005575): A function refers to the place in the cell where a gene product is active. It can be a general term such as nucleus or a specific term such as ribosome.

Particularly, The GO project is a collaborative work across many laboratories and controlled by the gene ontology Consortium (set of model organism and protein databases and biological research communities actively involved in the development and application of the Gene Ontology).

Hypergometric Test

If the bicluster we want to test its enrichment contains genes like [g_1; :::g_n]. The enrichment question is like this: Are there any GO terms that have a larger than expected subset of our bicluster genes in their annotation list? If so, these GO terms will give us insight into the

functional characteristics of our bicluster. The hypergeometric test calculates the probability of drawing r genes with a certain GO function from a sample of size k from a population of size n given that this GO function exists in fraction p in the population set of genes. The basic question answered by hypergometric test is as described by:
When sampling X genes (test set) out of N genes (reference set, either a graph or an annotation), what is the probability that x or more of these genes belong to a functional category C shared by n of the N genes in the reference set?.
The hypergeometric test, in which sampling occurs without replacement, answers this question in the form of P-value. Its counterpart with replacement, the binomial test, which provides only an approximate P-value, but requires less calculation time.

GO Enrichment Programs

There are various tools (web based and standalone applications) introduced to analyze GO term enrichment in a given genes set. Some of these tools have been developed by the GO Consortium such as AmiGO and OBO-Edit, while other tools have been developed outside the GO Consortium for use with GO ontologies such as BiNGO, GeneMerge, GOEAST and FuncAssociate . A comprehensive list of all these tools can be found at GO website (http://www.geneontology.org/GO.tools.shtml.).
The shortcoming of these programs is that you should to enter each bi/cluster manually and then count the enriched and unriched biclusters , which is consuming time and hard to do manually.