7-25-2011

# Analysis of Microarray Data with Directed Graphs

Nam Sy Vo

To the University Council:

The Thesis Committee for Nam Sy Vo certifies that this is the final approved version of the following electronic thesis: “Analysis of Microarray Data with Directed Graphs.”

_____

Vinhthuy Phan, Ph.D.
Major Professor

We have read this thesis and recommend its acceptance:

_____

Ramin Homayouni, Ph.D.

_____

Ebenezer O. George, Ph.D.

Accepted for the Graduate Council:

_____

Karen D. Weddle-West, Ph.D.
Vice Provost for Graduate Programs

# ANALYSIS OF MICROARRAY DATA WITH DIRECTED GRAPHS

by

Nam Sy Vo

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Bioinformatics

The University of Memphis

August 2011

# Dedication

I would like to dedicate this thesis to my dear parents, who I am always greatly indebted to, who are always ready to support me everything in my life with an unconditional love; to my dear younger brother, who always stands beside me from my childhood. They have been a great source of motivation and inspiration for me throughout my life.

## Acknowledgements

This thesis marked a completion of my two-year study at the University of Memphis. The thesis would not have been possible without the help from many people.

First of all, I would like to acknowledge Dr. Vinhthuy T. Phan, my major advisor, for his very useful advices. He helped me understand a lot of problems thoroughly. I thank him for spending hours of his valuable time with a terrific patience to support me solve problems, not only in this thesis but also in whole my work. I am deeply thankful for everything he has done for me.

My special thanks go to my committee members, Dr. Ramin Homayouni and Dr. Ebergener O. George for their patience as well as a lot of useful suggestions for my thesis. I would also like to thank Dr. Ramin Homayouni for his useful support during my study in the Department of Bioinformatics.

I would like to thank Dr. Thomas R. Sutter, who provided me an opportunity to work with him in a software project that is an important part in my thesis. It has been a pleasure to work with Ashutosh K. Pandey who explained me everything in his software that related to my work.

Last but not least, many thanks to my friends, classmates and teachers who helped me from the beginning. Especially, I am very grateful to Hai Nguyen and his family who supported me very much in my life during the time I studied here. Thank you very much.

## Abstract

Vo, Nam Sy. M.S. The University of Memphis. August 2011. Analysis of Microarray Data with Directed Graphs. Major Professor: Vinhthuy Phan, Ph.D.

This thesis studies various problems of analyzing microarray data with multiple treatments and multiple replicates using directed graphs. We start from a method that represents gene response as transitive directed acyclic graphs to extract meaningful information from microarray data (Phan et al., 2009). To extend this work, we simulate expressions of genes based on real data and make predictions about potential patterns of genes and their properties. From this, we predict whether such potential patterns are certain if given more replicates. Next, we used fold information which was not used in prior work to cluster genes share the same patterns and thus providing an additional layer of information. Finally, we created a web-based application to help other scientists try our method. The software assigns graphical patterns to genes and outputs images and textual data for further analyses.

## Table of Contents

## List of Figures

# Chapter 1

## Introduction

Gene expression analysis with DNA microarrays has become one of the mainstreams in the field of molecular biology. In living organisms, gene expression is the process that uses information from a gene to synthesize a functional gene product (protein or functional RNA). Gene expressions are often measured by DNA microarrays, one of the most popular technologies in this research. In a microarray experiment, expression levels of thousands of genes are simultaneously measured and the data are used to study the effects of drugs or diseases.[1]

While obtaining gene expressions is no longer a challenge in genomics research, analysis and interpretation of microarray data is still a major challenge (Subramanian et al., 2005). There are many methods proposed to analyze microarray data. Typically, the analysis of microarray data is divided into three steps: (1) selecting significant genes for further analyses, (2) analyzing these genes with clustering or classification methods and (3) interpreting gene clusters or classes to extract biological meaning from them (Phan et al., 2009). Most of current approaches often consider these steps separately and neglect to take advantage of results from the other steps. Therefore, comprehensive methods that take into account more than one step are expected to help understand more from the data.

### Coping with Multiple Treatment Microarray Data

In many cases, researchers have to work with gene expression data that involve many chemical compounds (or treatments) (Sutter et al., 2002; Sawers, Liu, Anufrikova, Hwang, & Brutnell, 2007). Moreover, because of biological variations in cells, people often need several sets of measurements of gene expressions (Yang, Yang, McIndoe, &

[1] http://www.bioinformaticstutorials.com/?p=8 - last visited 07/2011

She, 2003). Each set of measurement is called a replicate. Microarray data with multiple treatments and multiple replicates are often stored in several formats such as GCT, RES or PCL.[2] Figure 1 shows a gene expression dataset with 2 treatments and 4 replicates per treatment in GCT format.

| | | (Treatment group 1) | | | | (Treatment group 2) | | | |
|---|---|---|---|---|---|---|---|---|---|
| Probe Set | Gene information | CON1 | CON2 | CON3 | CON4 | TREAT1 | TREAT2 | TREAT3 | TREAT4 |
| 1369943_at | Rn.10 | 2177.696 | 1045.617 | 1941.677 | 1235.957 | 1691.027 | 1032.806 | 1688.829 | 1049.005 |
| 1367657_at | Rn.1000 | 8434.484 | 9746.999 | 7945.054 | 9150.292 | 11070.688 | 7135.596 | 11664.801 | 7243.845 |
| 1368270_at | Rn.10002 | 19.792 | 19.511 | 19.761 | 14.282 | 14.486 | 18.669 | 20.033 | 14.832 |
| 1372094_at | Rn.100021 | 996.149 | 867.428 | 1035.417 | 849.577 | 1071.943 | 1095.091 | 985.208 | 866.036 |
| 1399029_at | Rn.100022 | 102.423 | 118.124 | 77.588 | 107.665 | 103.208 | 130.816 | 87.883 | 103.508 |
| 1376849_at | Rn.100022 | 217.969 | 168.746 | 149.847 | 170.000 | 241.984 | 148.219 | 145.839 | 182.011 |
| 1397005_at | Rn.100042 | 6.006 | 6.366 | 6.387 | 6.543 | 6.351 | 7.211 | 6.320 | 7.459 |

*Figure* 1. A gene expression dataset with 1 control group and 1 treatment group.

In general, it requires further efforts to analyze and interpret gene expressions with multiple treatment microarray data (tens or even hundreds of chemical compounds). If there will not be enough replicates, it would be difficult to draw statistically reliable conclusion from the data (Pan & Le, 2002). That lack of replicates may lead to misleading/uncertain results.

**Analysis of microarray data with transitive directed acyclic graphs**

Phan et al. (2009) proposed a method for clustering multiple treatments gene expression data with transitive directed acyclic graph (tDAG). tDAGs are graphical representations of patterns of outcomes of statistical comparisons of gene expression profiles. Concretely, given a gene expression profile, perform all possible pairwise comparisons based on two-sided hypotheses we obtain a set of possible outcomes. These outcomes can be represented as a tDAG. Although based entirely on a statistical

[2] http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats - last visited 07/2011

construction, these outcomes reveal meaningful biological information about the tDAG that the genes fall into. It was shown that this method deals with multiple treatment microarray data better than other popular clustering methods such as hierarchical clustering or k-mean. This method has been applied and found useful in identifying and differentiating genes sharing similar functional pathways (Tran et al., 2009).

In this work, graphical patterns are determined using Wilcoxon rank-sum test to do pairwise comparisons for treatments, where:

- Vertices consist of the treatments $T_1$, $T_2$, …, $T_n$
- Edges represent the outcomes of Wilcoxon rank-sum tests:
  - "$T_1 \rightarrow T_2$" stands for the outcome $T_1 > T_2$ which means that gene is expressed significantly more under $T_1$ than under $T_2$.
  - "No edge between $T_1$ and $T_2$" stands for the outcome $T_1 \sim T_2$ which means that there is no statistical difference of expression of the gene under $T_1$ and $T_2$.

Each set of outcomes corresponds to an acyclic transitive closure with n vertices, that is, these patterns are transitive directed acyclic graphs. In additional, these graphs can be contractible or non-contractible. We say that a graph is contractible if (1) it is a complete graph or (2) it can be contracted to a complete graph that preserves all vertex-edge relationships in the original one. Phan, George, Tran, & Sutter (2008) showed that there is a relationship of the sample size problem with a property of directed graphs known as contractibility.

Figure 2 shows two graphical representations of gene response to 4 treatments (CON, TREAT1, TREAT2, and TREAT3) with a non-contractible graph (A) and a contractible graph (B). In particular, graph (A) shows a pattern with 1 control and 3 treatments where

gene is expressed significantly more under TREAT2 than under TREAT1, more under TREAT1 (TREAT2, TREAT3) than under CON and there is no statistical difference of expression of the gene under TREAT1 (TREAT2) and TREAT3. In graph (B), there is another edge from TREAT3 to TREAT1, which means gene is expressed significantly more under TREAT3 than under TREAT1. The left graph can be contracted to the right graph by grouping 2 equivalent vertices TREAT2 and TREAT3.

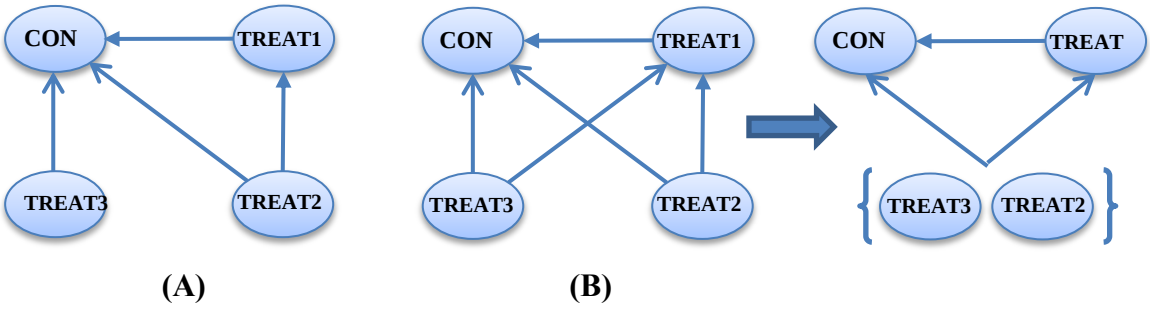


*Figure* 2. A non-contractible graph (A) and a contractible graph (B).

With such graphical presentation of gene responses, the analysis of microarray data is done through three steps: (1) selecting significant genes using Kruskal-Wallis with false discovery rate controlled; (2) determining the pattern of response of each gene using Wilcoxon rank sum tests to detect how the gene responds to all pairs of treatments; and (3) grouping genes with similar patterns into the same cluster. In addition, similar patterns are further placed into "meta" clusters. For future reference of this method, we call this method MPC (multiple pairwise comparisons). In the rest of this thesis, we extend this method to analyze further patterns as well genes in such patterns to extract meaningful information and make more reliable predictions.

**Content of Thesis**

The first problem studied is the contractibility of patterns when the number of replicates increases. The gene expression patterns obtained with more replicates might differ from each other considerably. Our objective is making predictions about probable patterns by simulating gene expressions based on real data with few replicates. We show that solving these problems leads us to meaningful results. It helps us predict whether potential pattern of genes are certain when there are more replicates. From this, we can quantify the relationship between the contractible patterns and sample size (i.e., number of replicates). It allows people predict how many samples needed for a non-contractible graph to become contractible, which help determine the necessary number of microarrays needed to get reliable results.

In the second problem, we consider the magnitude of gene expression data, which is ignored in the first problem. Although genes in patterns share common characteristics that reflect their responses with treatments, they are might different each other on their expression levels. Incorporating magnitude of expression data into the MPC method might provide us meaningful results. In the second problem, we use hierarchical clustering to analyze genes in patterns further using fold information between expression profiles of treatments. We also create a tree view of clusters to help people analyze further results.

The last problem in this thesis is building a software package for analyzing microarray data. This software is called mDAG. This web-based application allows users upload gene expression datasets. Analysis is done through steps in MPC method. User's requests are scheduled to run in such a way that it does not overload the server. The

pattern of response of any gene is represented visually. mDAG is implemented using Python, MySQL and web2py – a free open source framework for development of web-based applications.[3]

[3] http://web2py.com/ - last visited 07/2011

## Chapter 2

## Problem Definition

First, we present a formulation for the problem of predicting potential patterns from gene expression data using simulation. Second, we describe a method for clustering gene patterns based on fold information. Finally, we consider requirements of a tool for analyzing microarray data that implement the MPC method. We also consider some related problems for further discussions about this field of interest.

### Predicting Potential Patterns from Gene Expression Data

Phan et al. (2008) showed that, with large numbers of replicates, more contractible patterns will be observed hence we can get reliable conclusions. However, researchers often deal very few numbers of replicates (due to cost). In this situation, researchers might get uncertain results due to insufficient information. For example, Figure 3 (a) shows a non-contractible observed pattern when the number of replicates is insufficient. If we have a sufficient number of replicates, (a) might become either pattern (b), (c) or (d), which are contractible.

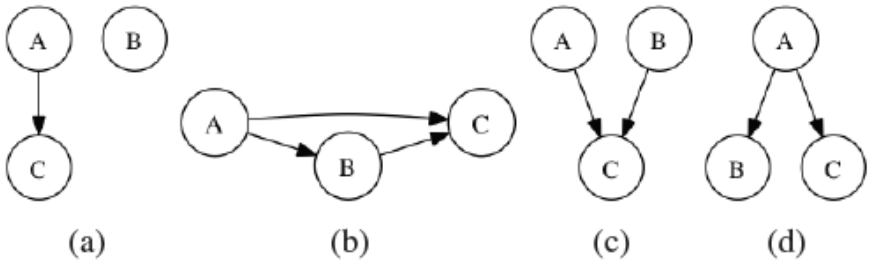


*Figure* 3. A non-contractible pattern (a) and its possible derived patterns (b, c, d).

The question is, which contractible patterns can a non-contractible pattern become to when more replicates are added? Formally, given a non-contractible pattern G, let $S_G$ be the set of all contractible patterns from which G can become. We want to calculate the probability $p_i$ that G will become $G_i \in S_G$ when replicates are sufficiently large. Determining exactly and efficiently $S_G$ or $|S_G|$ might be a hard problem because the numbers of ways to add more edges are exponential. However, solving this may help scientists to determine whether certain response patterns are reliable even when sample size is not sufficient.

One of the feasible ways to predict these is using synthetic data to produce more replicates and observe the results. By simulating synthetic gene expressions based on real samples (with few replicates), we can make predictions about patterns that might have been generated for genes with non-contractible patterns. We will have various models of simulation of gene expression with various assumptions including normal distribution of gene expressions as well as non-parametric methods; for example using methods similar to (Albers, Jansen, Kok, Kuipers, & Hijum, 2006).

In this thesis, the synthetic gene expression data has a Gaussian distribution. We calculate mean and standard deviation for this distribution from sample means and sample standard deviations of the real data. After that, we perform experiments on the data and make predictions based on the results.

**Clustering further genes in patterns based on their magnitudes**

As previously mentioned, one of remaining problems of the MPC method is the differences in magnitude of gene expressions. For example, consider two genes in Figure 4, although they belong to the same pattern with CON > TREAT1 > TREAT2, the

magnitude of CON is 10-fold the one of TREAT1 in gene A but only 2-fold the one of TREAT2 in gene B. This maybe leads to different properties between genes.

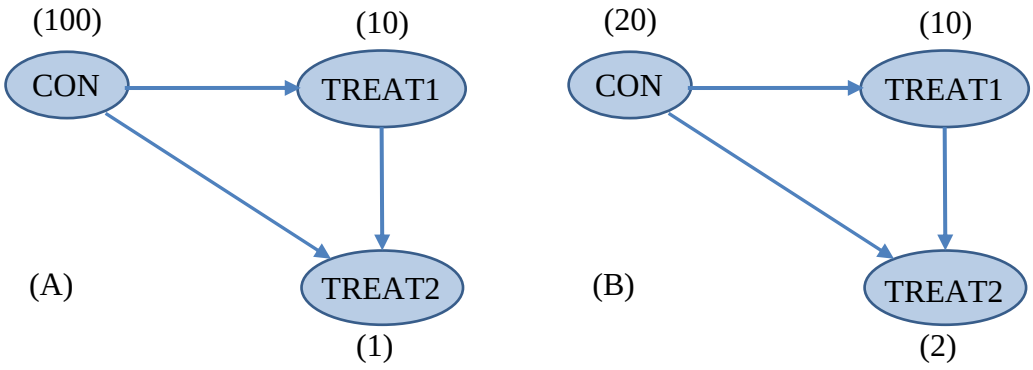


*Figure* 4. An illustration of two genes with the same graphical pattern but different in their magnitudes. The numbers in brackets represent the magnitudes of treatments.

The problem is how to incorporate fold information between gene expressions of treatments into MPC method. This leads to another approach where we consider patterns as weighted graphs instead of non-weighted graphs. In this model, each edge has a weight which is the fold-value between gene expression profiles of treatments. One of advantages of this approach that is we can classify further genes with the same patterns into subclasses, which might bring us another useful layer of information.

We implemented an algorithm based on hierarchical clustering with several metrics (Baxevanis & Ouellette, 2006; Deonier, Tavare, & Waterman, 2005; Gopal, Haake, Jones, & Tymann, 2006). We created a tree view of clusters to allow users visualize clustering results.

**Constructing a tool for analyzing microarray data using directed graphs**

As a web-based application, mDAG needs to take as input microarray data from users over the web. After that, it selects significant genes, performs statistical analysis, and

places them in appropriate patterns. Analysis is done in three steps: (1) significant genes are selected using Kruskal-Wallis with false discovery rate controlled; (2) the pattern of response for each gene is determined using Wilcoxon rank sum tests to detect how the gene responds to all pairs of treatments and (3) genes with similar patterns are placed into the same cluster. Similar patterns are further placed into "meta" clusters. The details of this process can be found in Phan et al. (2009).

To manage requests and results properly, mDAG needs a "Scheduler" program. Scheduler uses server resources effectively and schedules users' requests. Submitted requests are scheduled to run in a way that it does not overload the server. The Scheduler needs to call an "Executer" program, which is responsible for analyzing datasets. The Executer updates status of the requests and performs the MPC method.

Data sets must be a text file in GCT format [4] with two restrictions: (1) missing expression value is not allowed for expression data and (2) treatment groups are grouped together: the first group is Control group; the second group is the first treatment group; the third group is another treatment group; and so on. The software also needs to represent results in such a way that they are convenient for people to analyze further them as well as link them to external resources.

[4] http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats - last visited 07/2011

# Chapter 3

## Methods and Experiments

In this section, we present a method for generating synthetic gene expression based on real data and analyzing them with MPC method. We also describe experiments for predicting potential patterns from synthetic data. Second, we describe a method for calculating distance between genes for hierarchical clustering based on fold information. Last, we design a workflow for mDAG that satisfies all requirements in section 2.3.

### Generating synthetic gene expressions for predicting gene response patterns

As mentioned in previous section, there are various models of simulating gene expression with various assumptions including normal distribution of gene expressions as well as non-parametric methods. In this study, the assumption is synthetic gene expression data has a Gaussian distribution. Means and standard deviations for this distribution are calculated from the real data.

Suppose a gene has $n$ replicates for each treatment. We used these replicates to generate $k$ additional replicates for this treatment group with parameters ($\mu$, $\sigma$) that are the sample mean and standard deviation of the real replicates. Concretely, assume that we have a gene with non-contractible pattern $p$ from a real expression profile with $n$ replicates ($x_1, x_2 \ldots x_n$). The $p$ is obtained using Wilcoxon rank-sum test for all pairwise comparison. Do the following steps:

1. Generating a synthetic expression profile ($x_{n+1}, x_{n+2} \ldots x_{n+k}$) for $k$ more replicates. Assume that the synthetic data set has normal distribution with parameters ($\mu$, $\sigma$) are the sample mean and the sample standard deviation of the real data set.

2. Using Wilcoxon rank-sum test for the data ($x_1$, $x_2$ … $x_n$, $x_{n+1}$, $x_{n+2}$ … $x_{n+k}$), we obtain a pattern $p_1$.
3. Similarly, for another synthetic expression profile ($x'_{n+1}$, $x'_{n+2}$ … $x'_{n+k}$), we obtain another pattern $p_2$, and so on.

From this result, we can determine a list of observed patterns and their frequencies (for every sample) for all synthetic replicates as illustrated in Figure 5. For each gene, with *M* samples, we can obtain the list of patterns $p_1$, $p_2$ … $p_m$ for each more synthetic replicate. Some patterns among them probably are the same, therefore the number of distinct patterns often less than *M*. Notice that in one experiment, the synthetic samples for previous replicates are retained for next samples. These results can be represented as matrices.

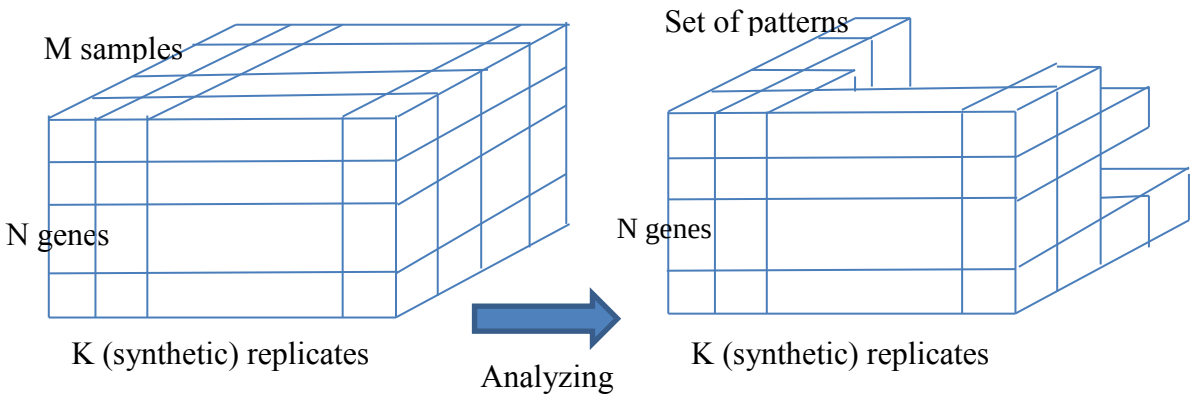


*Figure* 5. An illustration of observed patterns and their frequencies.

Based on this method, we can design an experiment as the followings:

i. Determining the pattern for each gene with real data. Call them initial patterns for each gene. Some of them are contractible, the others are non-contractible.

ii. Generating synthetic data with 30 more replicates for all genes with 100 samples for each more replicate. In each sample, the synthetic samples for previous replicates are retained for next replicates.

iii. Using Wilcoxon rank-sum test to obtain patterns for each gene for each sample and each replicate.

iv. Collecting all data as frequency matrices for observed patterns. Representing patterns in frequency charts to consider their behavior.

After all, we can perform necessary calculations on this empirical data to produce useful predictions.

**Calculating distance between genes for hierarchical clustering**

In this section we describe a method to calculate distance between genes in pattern based on fold information of gene expression data. First we need to compute the "representative value" of treatments and then we can calculate the fold-value between treatments. The distance between 2 genes is calculated based on their fold-values.

If there are k treatments (including control) with several replicates, then each gene can be represented as a vector of k treatments: $G_i = (T_1, T_2 \ldots T_k)$ where each treatment is a vector with *n* values (for *n* replicates): $T_i = (r_1, r_2 \ldots r_n)$. Then, the "representative value" of each treatment is computed as *normalized norm* $L_1$ (mean) of $T_i$[5]. From this, we can compute the fold-value between treatments as follow: $F_i = \{f(T_1, C), f(T_2, C), f(T_1, T3), f(T_2, T3)\}$ where *f* is *fold-function*. The fold-value indicate the differentiate genes in the same pattern. Here we ignore edges between unrelated vertices.

Figure 6 illustrates the representative value for expression profiles of treatments and fold-value between them for a gene with its pattern. With this distance, we can apply a

[5] http://mathworld.wolfram.com/L1-Norm.html - last visited 07/2011

clustering method to analyze genes in patterns further. We elected to use hierarchical clustering with several metrics and linkage criteria.

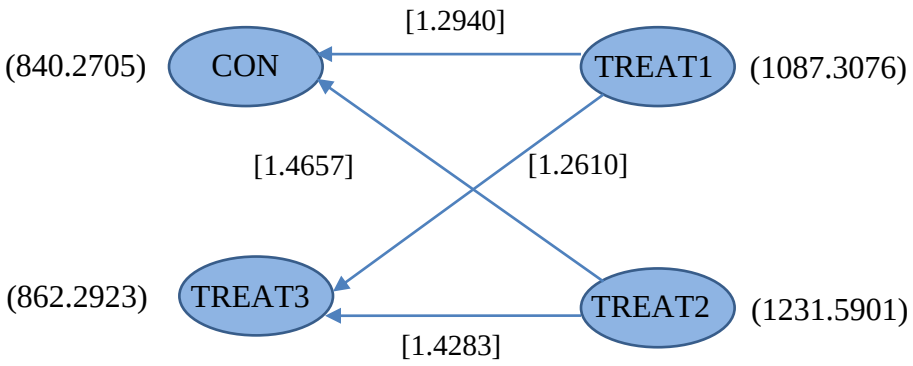


*Figure* 6. Representative value for expression profiles of treatments and fold-value between them. The numbers in brackets represent the magnitudes of treatments. The numbers in square brackets represent the fold-value between treatments.

**Designing the software: mDAG**

In this section, we design a Scheduler (Figure 7) that scheduled submitted requests to run in such a way that they do not overload the server. There are often many requests at the same time and Scheduler need to allow only some processes to run. Numbers of processes are determined based on existing resource. In this software, we set a fixed number of requests as a threshold and allow more requests if resource still enough. The following is the main responsibilities of the Scheduler:

i. When a request is sent to server, the Scheduler check the running requests and determine whether allowing more requests to run. Requests are analyzed respectively in incoming-time order.

ii. Status of requests includes "Waiting" (uploaded, not analyze yet), "Running" (is being analyzed), "Completed" (analyzed, there is result) or "Error" (there are errors

when analyzing the data, for example, no gene is significantly differentially expressed or the data format is not conformable with the program).

iii. Call Executer program to analyze the dataset. Executer update status of the request to Running, Completed or Error, perform MPC method to produce results. The Executer need to call Scheduler after finish a request.

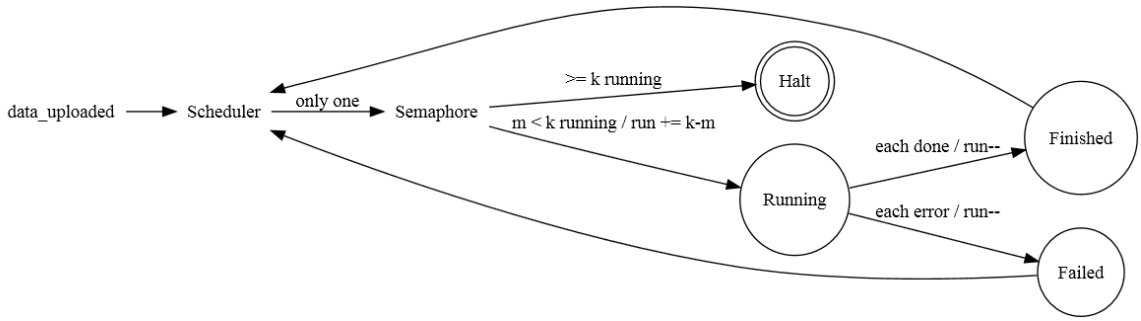


*Figure* 7. Sequence diagram of the Scheduler program in mDAG

The MPC clustering methods had been implemented in Phan et al. (2009). With some modifications, it is incorporated into mDAG as a part of Executer. More detail about MPC method can be found in Phan et al. (2009). Some codes (in Python) for this method can be found at http://binf2.memphis.edu/ashutosh/pythoncodes.html - last visited 07/2011.

# Chapter 4

# Results

## Predictions about Potential Patterns

We implement a program in Python for obtaining the list of observed patterns (with synthetic data) and their frequencies for all replicates and samples. Another module is used to analyze them for obtaining necessary results. We use a dataset from Phan et al. (2009), which has 12906 genes, 4 treatments (include control) with 5 real replicates per treatment. After generating synthetic data with 30 more replicates for all genes with 100 samples for each more replicate, the genes in such patterns are analyzed to determine the pattern for each sample and each replicate. Finally, we collect all data as frequency matrices for observed patterns and analyze them for further studying.

Figure 8 shows an analysis on a gene with initial non-contractible pattern from which several contractible patterns are predicted. The total of probability of all patterns (shown in different colors) per replicate is 1. Larger portions mean those patterns are more likely observed from the non-contractible pattern. After using about 20 synthetic replicates, 3 contractible patterns emerge as most likely from a non-contractible pattern.

Figure 9 shows average entropy of genes. The entropy for each replicate was calculated from the frequencies of observed patterns. Entropy decreases when we have more replicates show that divergence of patterns decreases.

From Figure 8 and Figure 9 we can observe the trend of changes of response of a gene in a non-contractible pattern when the sample size increases.

Last, Figure 10 shows the entropy boxplot for each more replicate of this gene.

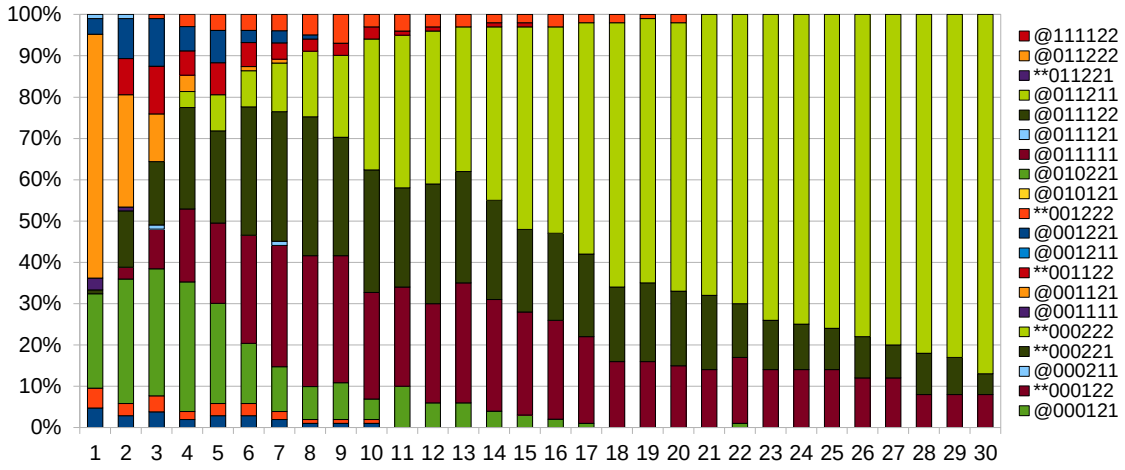


*Figure* 8. An analysis on a gene with non-contractible pattern.

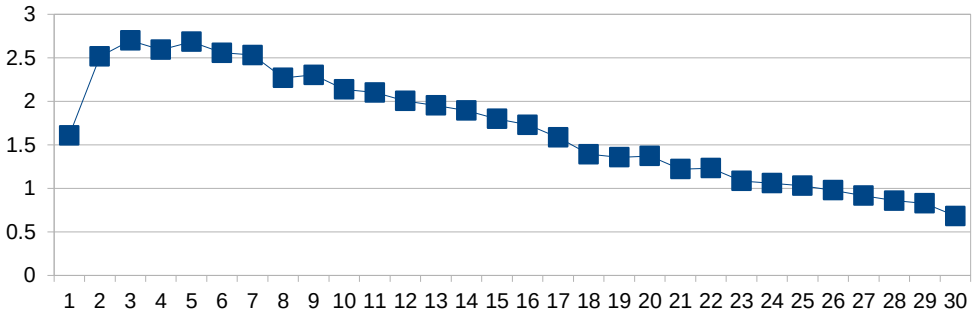


*Figure* 9. Entropy plot (on average) for the analysis in Figure 8.

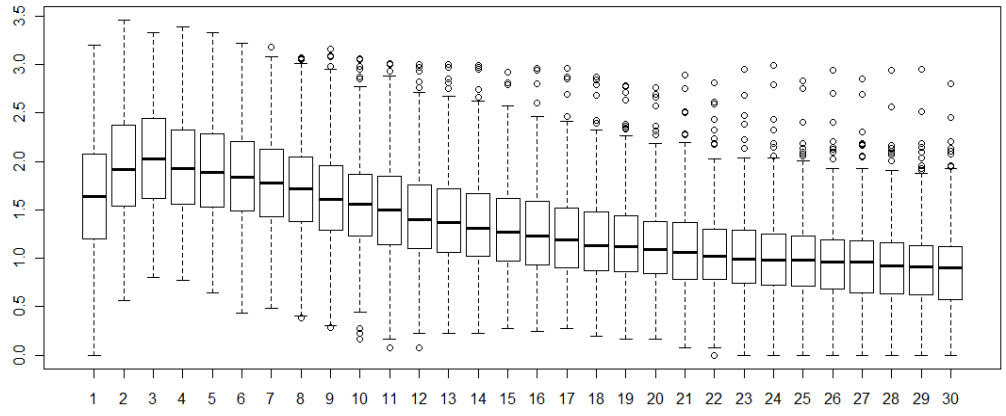


*Figure* 10. Entropy boxplot for the analysis in Figure 8.

In the rest of section, we present predictions about the patterns. These predictions based on calculating entropy and/or probability of potential patterns.

**Synthetic replicates help predict potential patterns of a gene.** For genes in both contractible and non-contractible patterns, several patterns are most likely to be observed. Entropies are reduced when more replicates are added (Figure 11).

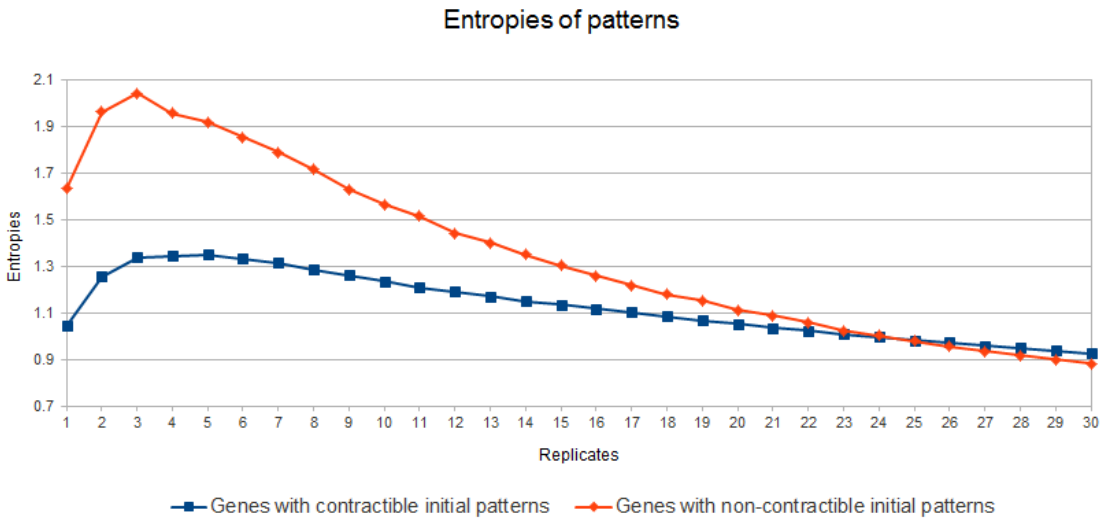


*Figure* 11. Average entropies of patterns of all genes for 30 more replicate.

**Predicted patterns are likely contractible.** Most of potential patterns are contractible patterns: probabilities of contractible patterns increase to 1 (Figure 12). Because of the result from section 4.1.1, potential patterns converge to several ones, so when we have more replicate genes are distributed into several contractible patterns.

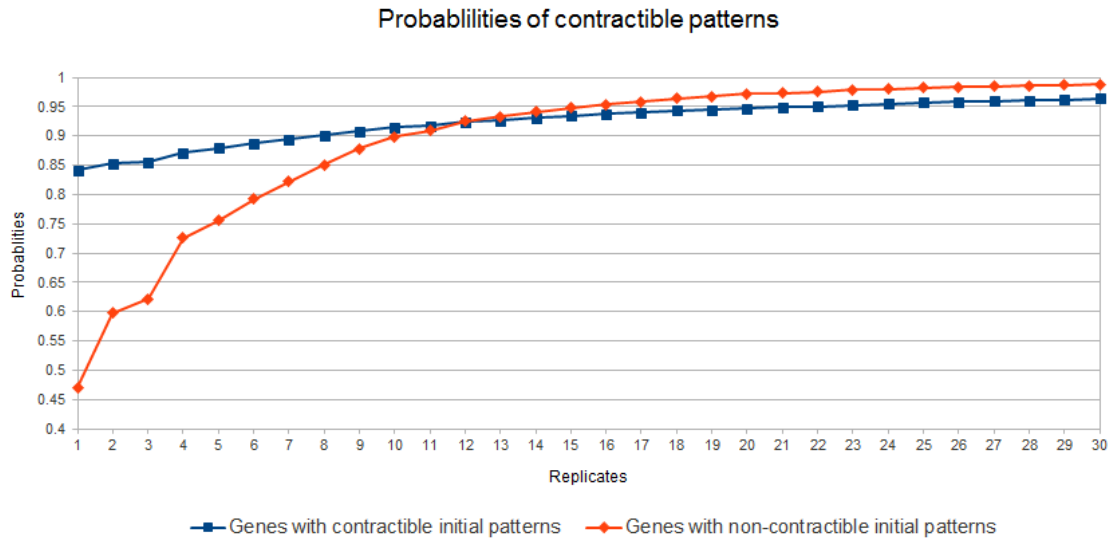


*Figure* 12. Average probabilities of contractible patterns for 30 more replicate.

**Predicted patterns are likely to be extensions of initial patterns.** Most of potential patterns are supper patterns of the initial patterns (with 5 real replicates): probabilities of contractible patterns increase to 1 (Figure 13). Again, because of the result from section 4.1.1, potential patterns converge to several ones, so when we have more replicates genes were distributed into several supper patterns (patterns that are extensions of initial pattern).

Moreover, because of the result from sections 4.1.2 and 4.1.3, predicted patterns are likely to be contractible extensions of initial patterns, so when we have more replicates genes were distributed into several contractible supper patterns.

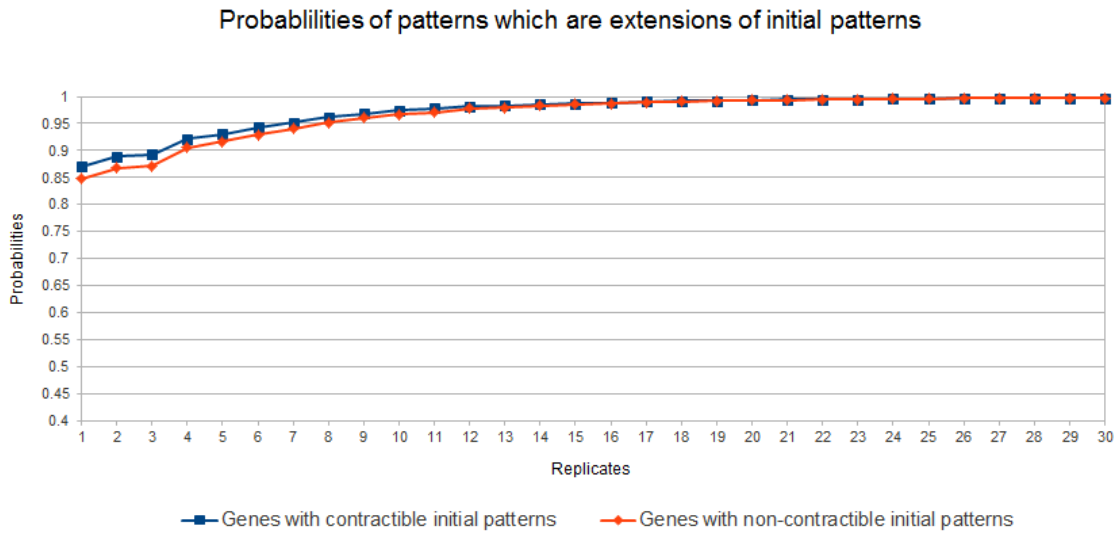


*Figure* 13. Average probabilities of patterns which are extensions of initial patterns of all genes for 30 more replicate.

Probabilities of potential contractible patterns, extensions of initial patterns and contractible extension of initial patterns are very close each other (Figure 14 - for contractible genes and Figure 15 - for non-contractible genes)

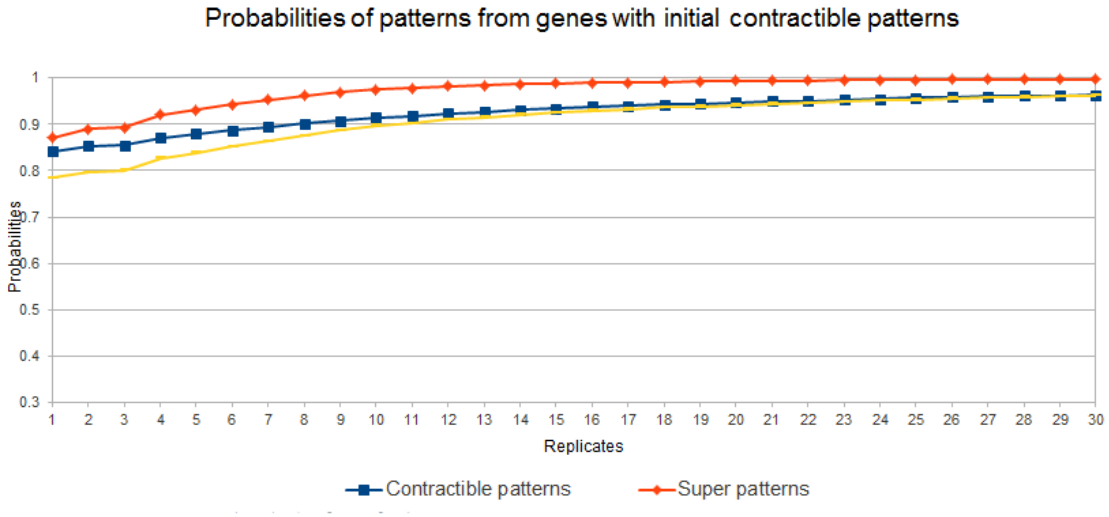


*Figure* 14. Average probabilities of patterns from genes with initial contractible patterns for 30 more replicate.

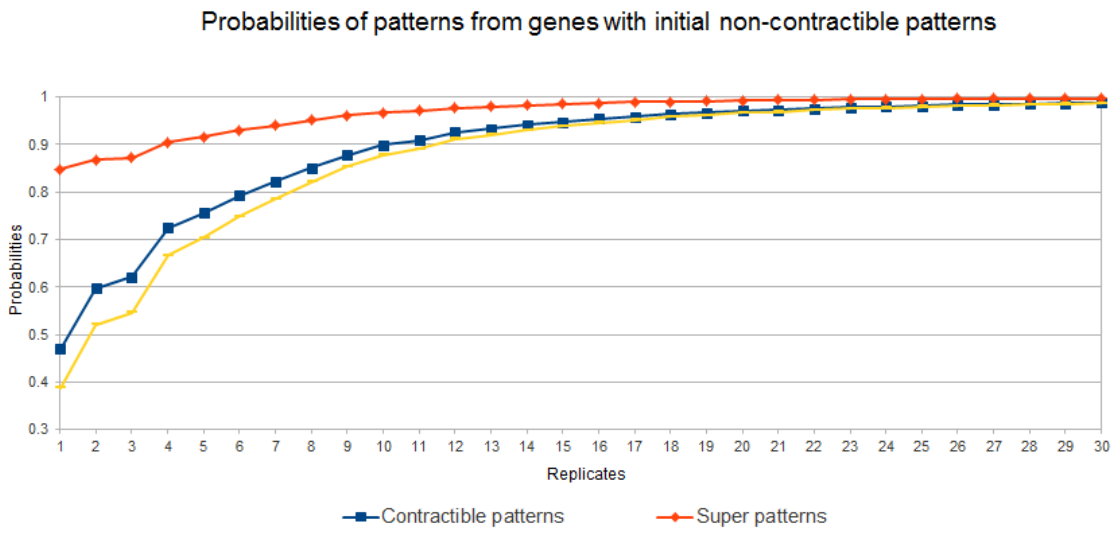


*Figure* 15. Average probabilities of patterns from genes with initial non-contractible patterns for 30 more replicate

**Potential patterns are the same with initial patterns in contractible genes than non-contractible genes (Figure 16).**

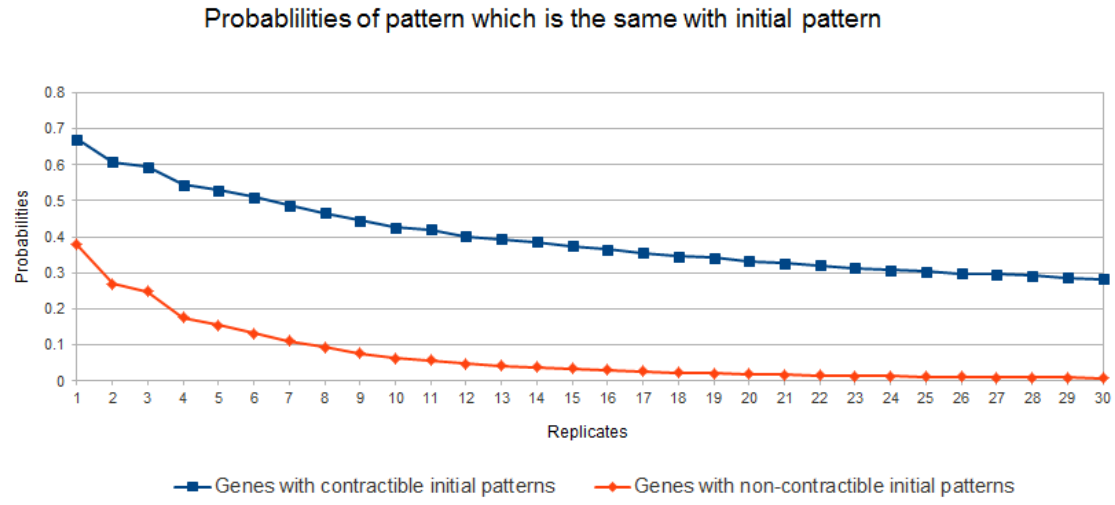


*Figure* 16. Average probabilities of patterns which is the same with initial patterns for 30 more replicate

In particular, most potential patterns of genes with complete patterns are the same with initial patterns, their probabilities very close to 1 (Figure 17).

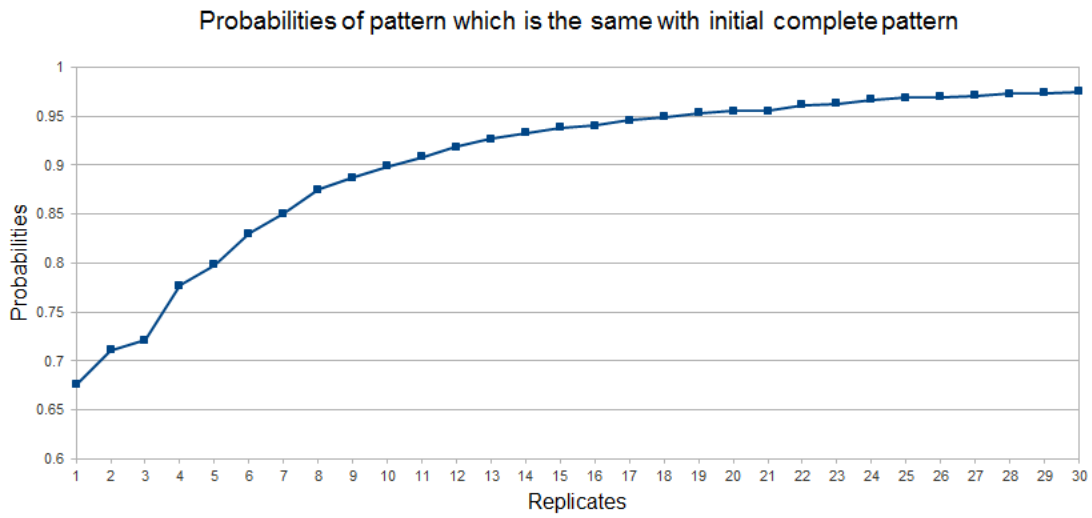


*Figure* 17. Average probabilities of patterns which is the same with initial complete patterns for 30 more replicate

**Clustering Results for Genes in Patterns**

To cluster genes in patterns, we implement hierarchical clustering in Python with several metrics and linkage criteria. About metrics, we use the Euclidean distance and Pearson correlation coefficient. We implement single-linkage/complete linkage and average linkage. All criteria are implemented in such a way that they compute distance update step in running-time O(1). The distance formula for this algorithm is described in section 3.2.

About data structure, we store each node as a Vector and maintain a "count" field in each tree node, which holds the number of leaf nodes which belongs to it in hierarchical tree. To improve running-time, we store only lower triangular part of distance matrix.

The result of clustering is visualized as a tree with labels to convenient for analyzing them further. The labels include ID of derived clusters and number of genes in them, name and ID of genes in input data.

To convenient for manipulating on clusters, we implement functions that allow user:

- Draw tree view into an image file.
- Label nodes with necessary information.
- Extract a set of clusters with distance less than a given value.
- Extract a given number of clusters.
- Return all elements of a cluster

Figure 18 represents a result of clustering of 10 genes in pattern 222221 with 4 treatments and 5 replicates per treatment, using Euclidean distance with average linkage.

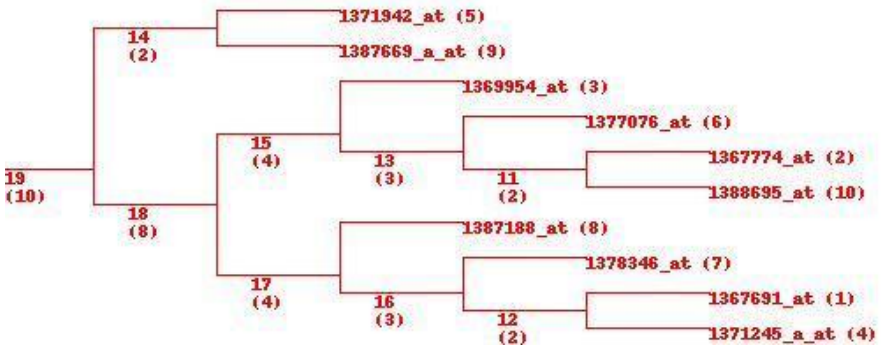


*Figure* 18. A result for clustering 10 genes in the same pattern
Inner nodes: the upper number is cluster's id and the lower number (in brackets) is total number of genes in the cluster. Leaf nodes: gene's name and gene's id (in brackets) in the input dataset.

Figure 19 represents result of clustering 168 genes in pattern 122221 with 4 treatments and 5 replicates per treatment, using Euclidean distance and average linkage. We can see some groups of genes are relatively separate. From this, there are some remarkable observations:

i. Effect of number of replicates: clustering vary through number of replicates (from non-contractible pattern to contractible pattern).

ii. Effect of metrics: results with Euclidean distance differ much from Person correlation coefficient.

iii. Effect of linkage criteria: results with average linkage (and complete linkage) differ much from single-linkage.

From the clustering result, we can extract groups for analyze further.

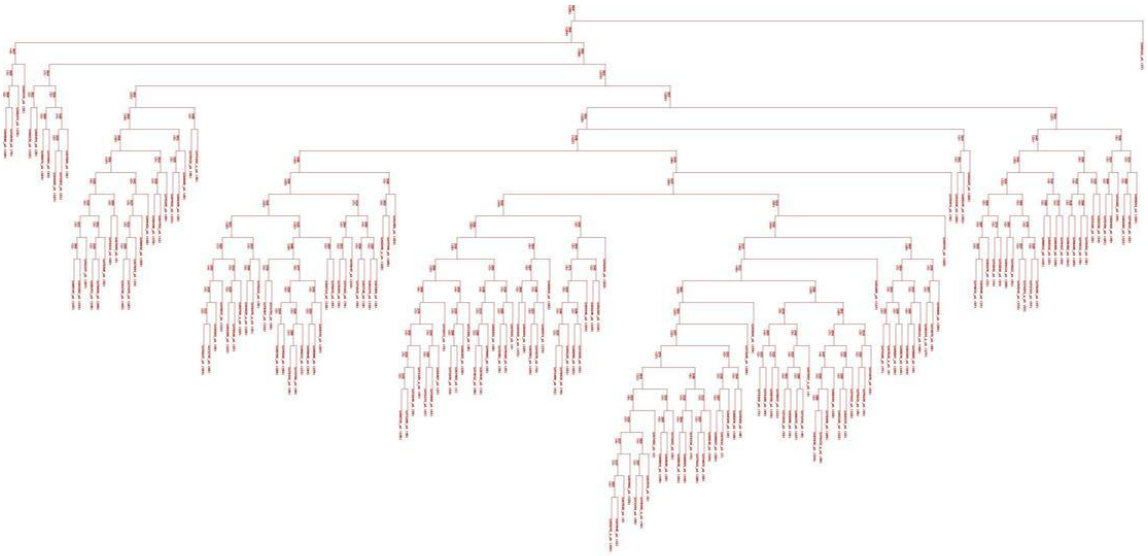

*Figure* 19. A result of clustering of genes in a pattern with separated groups of genes.

**Features of mDAG**

The application is implemented using Python, MySQL and web2py. User can use one of the sample data sets that are provided with the application to submit a request for analysis or use their own properly-formatted dataset.

Users can choose significance level p-value for the analysis (default value is 0.05). They can choose number of replicates for all treatments of the dataset. Finally, user can enter description of the request if needed (more about dataset, parameters or others).

After a request is finished, the user can check information about the requests and results, including dataset file and request description, status of the request, request upload time, analyzing start time, and finish time.

A request description includes all user information uploaded: number of probe sets, number and name of treatments with replications, significant level, replication number for each treatment and request description (if it exists). Users can download datasets.

Status of the request is one of the following: Waiting, Running, Completed, and Error. If status is “Completed”, users can check the analyzing result. If status is “Error”, users can re-run the corresponding dataset. Users also can re-run the dataset for dataset with “Completed” status.

Finally, users can remove the request. With "Waiting" or "Error" status, the system removes the corresponding dataset. With "Running" status, the system removes the corresponding dataset and stops the analyzing process. With "Completed" status, the system removes both the corresponding dataset and analyzing result.

# Chapter 5

# Discussion

## Simulation Method for Predicting Potential Patterns

Simulated data helps us to find out useful results that are conformable with theoretical predictions. However, there are two problems we currently face with this approach.

First, the simulation method takes much time to run. In many cases, there are a large number of samples as well as replicates need to test, hence the complexity of algorithms is very important. We exploited some techniques to improve its performance. For example, we use dynamic programming for computing exact p-values of Wilcoxon rank-sum test based on (Ross, 2002) (during the process of determining graphical patterns) and show that it is effective. Nevertheless, simulating synthetic expressions of several thousand genes can take up much time on regular desktops. For example, it takes two more than a week of computing to perform even with only 100 samples and 30 replicates. We exploited high performance cluster to improve the runtime but it still need to improve more, such as using parallel program.

Second, determining appropriate amount of synthetic replicates to generate for each gene is quite difficult. Generating too many synthetic replicates can be both computationally expensive and causing over-fitting results. This leads us to another problem: design computational efficient methods for selecting the right amount of synthetic replicates to predict probable contractible patterns from non-contractible patterns. However, this arises some computational questions that are non-trivial.

There are some interesting theory questions in this problem. First, can we predict observable patterns (or the probabilities of them) for more replicates without synthetic

data? For example, can we construct a mathematical formula to determine $S_G$ or $|S_G|$ for a given graph G efficiently? Although this question possible efficiently determined, the similar questions comparing G and another pattern G' can be computationally hard (Phan, 2010). For example, given a non-contractible pattern G and another pattern G', it is NP-hard to compute the minimum distance between a member of $S_G$ and a member of $S_{G'}$ (Blin et al., 2007; Fu & Jiang, 2007). What this implies is that given two observed patterns, which happen to be non-contractible, it is computationally hard to find out their closest distance if we were to have sufficient replicates.

**Incorporating Traditional Clustering into Current Method**

Besides the hierarchical clustering in this thesis, we developed a program to implement k-mean method for clustering genes in patterns. These methods were implemented with several distance measures and linkage criteria. One of the next works is incorporating this method into MPC method. We hope that it can bring us a considerable improving for the results from MPC method.

**Develop the Next Version of mDAG**

In the future we plan to integrate the results from the simulation method as well as some other software into mDAG. From this, users can make predictions with their datasets that have few replicates. The results will be presented on a web-based interface so that people can study them further.

# Chapter 6

## Conclusion

We showed that the simulation is useful to quantify the relationship between the contractible patterns and sample size (i.e., number of replicates or number of microarrays needed to get reliable results). By simulating synthetic gene expressions based on real data with few replicates, we can predict the potential patterns of a given gene with limited number of replicates. This may help scientists to determine whether certain response patterns are reliable even when sample size is not sufficient. Moreover, by analyzing the empirical datasets from synthetic data, we can predict how much more samples needed for non-contractible graphs to become contractible. From this point, we showed that there is a relationship between "contractible" patterns and the necessary number of replicates needed to get reliable results. This may help using a necessary number of microarrays.

Besides the variance of response that is exploited in the MPC method, we also consider the magnitude of gene expression. We use hierarchical clustering to cluster further genes in patterns where the distance is the fold information between expression profiles of treatments. The result from this method can be used to study further genes in patterns as well as improve the MPC method. We showed that this might produce another meaningful layer of information. We also create a tree view of clusters with support methods to convenient for analyzing further results.

As an application of the method, the mDAG software was constructed to provide a tool for user analyze microarray data with multiple treatments. It implements the MPC method that analyzes microarray data with multiple treatments and multiple replicates, using directed graphs. As a web-based application, mDAG allows users to upload

microarray data in GCT format through a web interface. From this data, the application performs calculations to assign graphical patterns to genes and outputs images and textual data for further analyses. mDAG is implemented using Python, MySQL and web2py.

In the future, we plan to analyze further properties of observed patterns from synthetic data as well as perform analysis with datasets which have more treatments. (Baldi, & Brunak, 2005; Gopal et al., 2006) provided some useful methods. Some other sources, such as GO [6] or GeneMania [7] can be the useful information sources. We also plan to study theoretical aspects of the problem, including the predictions of potential patterns.

[6] The Gene Ontology project: http://www.geneontology.org/ - last visited 07/2011

[7] The GeneMANIA project: http://genemania.org/ - last visited 07/2011

## References


Albers, C. J., Jansen, R. C., Kok, J., Kuipers, O. P., & Hijum, S. A. van (2006). Simage: simulation of dna-microarray gene expression data. *BMC Bioinformatics*, 7:205. doi: 10.1186/1471-2105-7-205.

Alon, N., & Spencer, J. H. (2008). *The Probabilistic Method* (3rd ed.). New Jersey: John Wiley and Sons.

Augen, J. (2004). *Bioinformatics in the Post-Genomic Era: Genome, Transcriptome, Proteome, and Information-Based Medicine* (1st ed.). Addison-Wesley.

Baldi, P., & Brunak, S. (2001) *Bioinformatics – The Machine Learning Approach* (2nd ed.). Massachusetts: The MIT Press.

Baxevanis, A. D., & Ouellette B. F. F. (Eds.) (2005). *Bioinformatics – A Practical Guide to Analysis of Genes and Proteins* (2rd ed.). John Wiley & Sons.

Blin, G., Blais, E., Hermelin, D., Guillon, P., Blanchette, M., & El-Mabrouk, N. (2007). Gene maps linearization using genomic rearrangement distances. *J Comput Biol, 14*(4), 394–407. doi: 10.1089/cmb.2007.A002.

Brinkmanm, G., & McKay, B. D. (2002). Posets on up to 16 points. *Order, 19,* 147–179.

Datta, S., & Datta, S. (2006) Evaluation of clustering algorithms for gene expression data. *BMC Bioinformatics, 7*(Suppl 4), S17. doi:10.1186/1471-2105-7-S4-S17

Deonier, R. C., Tavare, S., & Waterman, M. S. (2005). *Computational Genome Analysis – An introduction*. New York: Springer Science+Business Media.

Dudoit, S., Shaffer, J. P., & Boldrick, J. C. (2003) Multiple Hypothesis Testing in Microarray Experiments. *Statistical Science, 18*(1), 71-103.

Fu, Z. & Jiang, T. (2007). Computing the breakpoint distance between partially ordered genomes. *Journal of Bioinformatics and Computational Biology, 5*(5), 1087–1101.

Goeman, J. J., & Mansmann, U. (2008) Multiple testing on the directed acyclic graph of gene ontology. *Bioinformatics, 24*(4), 537–544. doi: 10.1093/bioinformatics/btm628.

Gopal, S., Haake, A., Jones, R. P., & Tymann, P. (2008). *Bioinformatics – A Computing Perspective* (1st ed.). McGraw-Hill Higher Education.

Gronau, I., & Moran, S. (2007). Optimal Implementations of UPGMA and Other Common Clustering Algorithms. *Journal Information Processing Letters, 104*(6)

Hulshizer, R. & Blalock, E. M. (2007). Post hoc pattern matching: Assigning significance to statistically defined expression patterns in single channel microarray data. *BMC Bioinformatics 8,* 240.

Kerr, G., Ruskin, H. J., Crane, M. & Doolan, P. (2008). Techniques for clustering gene expression data. *Computers in Biology and Medicine, 38*(3), 283–293. doi: http://dx.doi.org/10.1016/j.compbiomed.2007.11.001.

Pan, W., Lin, J., & Le, C. T. (2002). *How many replicates of arrays are required to detect gene expression changes in microarray experiments? A mixture model approach. Genome Biology, 3*(5).

Phan, V., George, E. O., Tran, Q. T., & Sutter, T. (2008). Toward a combinatorial approach to the sample size problem in multiple-treatment microarray studies. *2008 International Conference on Bioinformatics and Computational Biology (BIOCOMP 2008)*, 175–181.

Phan, V., George, E. O., Tran, Q. T., Goodwin, S., Bodreddigari, S., & Sutter, T.R. (2009). Analyzing microarray data with transitive directed acyclic graphs. *Journal of Bioinformatics and Computational Biology, 7*(1), 135–156.

Ross, M. S. (2002). *Simulation* (3rd ed.) San Diego: Academic Press.

Sato, K., Mituyama, T., Asai, K., & Sakakibara, Y. (2008). Directed acyclic graph kernels for structural rna analysis. *BMC Bioinformatics, 9*, 318–318. doi: 10.1186/1471-2105-9-318.

Sawers, R. J. H., Liu, P., Anufrikova, K., Hwang, J. T. G., & Brutnell, T. P. (2007). A multi-treatment 15 experimental system to examine photosynthetic differentiation in the maize leaf. *BMC Genomics 8*:12. doi: 10.1186/1471-2164-8-12

Shen, Y., Sun, W., & Li, K. (2010). Dynamically weighted clustering with noise set. *oinformatics, ),* 26(3), 341-347. doi: 10.1093/bioinformatics/btp671

Subramanian, A., Tamayo B., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, … Mesirov, J. P. (2005). Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *PNAS, 102*(43). doi: 10.1073/pnas.0506580102

Sutter, T. R., He, X. R., Dimitrov, P., Xu, L., Narasimhan, G., George, E. O., … Kensler T. W., (2002). Multiple comparisons model-based clustering and ternary pattern tree numerical display of gene response to treatment: Procedure and application to the preclinical evaluation of chemopreventive agents. *Molecular Cancer Therapeutics. 1*(14), 1283–1292.

Tran, Q. T., Xu, L., Phan, V., Goodwin, S., Rahman, M., … Sutter, T. R. (2009). Chemical genomics of cancer chemopreventive dithiolethiones. *Carcinogenesis, 30*(3), 480-486.

VanderWeele, T. J., & Robins, J. M., (2007. Directed acyclic graphs, sufficient causes, and the properties of conditioning on a common effect. *American Journal of Epidemiology, 166*(9), 1096–1104. doi: 10.1093/aje/kwm179.

Yang, M., Yang, J., McIndoe, R., & She, J. (2003). Microarray experimental design: power and sample size considerations. *Physiology Genomics, 16*(1), 24–28. doi: 10.1152/physiolgenomics.00037.2003

Yeung, K. Y., & Ruzzo, W. L. (2001). Principal component analysis for clustering gene expression data. *Bioinformatics, 17*(9), 763–774. doi: 10.1093/bioinformatics/17.9.763.

# Appendices

## Appendix A. More about simulated data

Figure 1A shows a number matrix for observed patterns for through replicates of one non-contractible pattern (000011) with synthetic data. This pattern has 6 genes. There are 30 more synthetic replicates (the columns) and 25 observed patterns (the rows) for all replicates. Many of them "disappear" when the number of replicates is enough large. When the numbers of replicates come up to about 20, four of them, that all are contractible patterns, dominate the others. In the figure, the patterns with ** mark are contractible while the ones with @ mark are non-contractible.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **000000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| **000001 | 65 | 115 | 116 | 131 | 135 | 147 | 150 | 149 | 158 | 150 | 149 | 144 | 137 | 130 | 118 | 118 | 119 | 112 | 110 | 113 | 107 | 106 | 105 | 98 | 91 | 87 | 91 | 90 | 89 | 89 |
| **000002 | 0 | 13 | 28 | 44 | 58 | 67 | 90 | 97 | 122 | 143 | 149 | 169 | 181 | 196 | 210 | 215 | 228 | 235 | 246 | 246 | 261 | 260 | 262 | 275 | 284 | 298 | 301 | 303 | 305 | 310 |
| @000011 | 384 | 251 | 234 | 161 | 139 | 111 | 94 | 83 | 56 | 47 | 38 | 23 | 22 | 18 | 16 | 14 | 8 | 8 | 4 | 4 | 2 | 2 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| **000012 | 73 | 164 | 174 | 222 | 233 | 248 | 244 | 253 | 244 | 240 | 243 | 244 | 238 | 233 | 231 | 226 | 218 | 220 | 213 | 208 | 202 | 203 | 205 | 196 | 195 | 184 | 174 | 173 | 170 | 170 |
| **000022 | 0 | 0 | 0 | 2 | 4 | 5 | 4 | 8 | 10 | 14 | 16 | 15 | 17 | 22 | 24 | 25 | 26 | 24 | 26 | 28 | 27 | 29 | 28 | 29 | 28 | 30 | 33 | 33 | 36 | 31 |
| **000111 | 25 | 13 | 13 | 8 | 6 | 4 | 4 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @000112 | 3 | 3 | 7 | 4 | 3 | 4 | 2 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **000122 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @001011 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @001012 | 2 | 4 | 2 | 4 | 6 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **001022 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @001112 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **001122 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **100001 | 6 | 6 | 5 | 8 | 7 | 4 | 5 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **100002 | 0 | 1 | 1 | 1 | 3 | 2 | 2 | 2 | 4 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @100011 | 12 | 14 | 6 | 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @100012 | 10 | 7 | 6 | 5 | 1 | 3 | 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @101001 | 3 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @101011 | 5 | 1 | 2 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **101012 | 5 | 6 | 3 | 0 | 0 | 1 | 1 | 2 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @101022 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @101111 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @101112 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| @111111 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure* 1A. Number of observed patterns with 100 synthetic samples per gene

The number matrices are converted to probability matrices and then transmitted to a data analysis module. These empirical data are analyzed and their results are put into Excel to produce necessary charts.

## Appendix B. More about hierarchical clustering

We implemented hierarchical clustering in Python with several metrics and linkage criteria. We also use Python packages numpy [8] and PIL/matplotlib [9].

* Metrics:

- Euclidean distance:

L2Dist = sqrt(sum(({v1}-{v2})**2)) # numpy-based style formula

- Pearson correlation coefficient:

Pcc = sum([({v1}-x1)/s1]*[({v2}-x2))/s2])/(n-1)) # numpy-based style formula

* Linkage criteria:

- Single-linkage and Complete linkage

d(Ck,Cl)) = min[d(Ci,Cl), d(Cj,Cl)]

d(Ck,Cl)) = max[d(Ci,Cl), d(Cj,Cl)]

- Average linkage:

d(Ck,Cl) = [d(Ci,Cl)*|Ci| +d(Cj,Cl)*|Cj|] / (|Ci|+|Cj|)  (|Ck|=|Ci|+|Cj|).

These formulas allows computes distance update step in running time O (1) (Gronau and Moran, 2006).

* Data structure:

- We store each node as a Vector and maintain count field in each tree node, which equals the number of leaf nodes belong to it.

- We store only lower triangular part of distance matrix.

[8] http://numpy.scipy.org/ - last visited 07/2011

[9] http://matplotlib.sourceforge.net/index.html - last visited 07/2011

* Complexity of algorithms:

    - Update distance: O (1)

    - Find closet element pair: O (n^2).

* Functions for manipulating on cluster results:

    - Codes for drawing tree view, labeling inner nodes, extracting clusters and printing clusters in the form of text-tree are based on work by Jan Erik Solem.[10]

[10] http://www.janeriksolem.net/2009/04/hierarchical-clustering-in-python.html - last visited 07/2011

## Appendix C. More about mDAG

### Dataset format

*General information:*

- Data must be stored in a text file.
- Columns are separated by tabs. The best way to do this is select appropriate columns in your Excel files and save them as text.
- The first row is the header that describes each column.
- Each row, starting from the second, corresponds to each gene/probe.

*Columns:*

- First column: Probe Set Id
- Second column: Gene Symbol
- Third column: UniGene Id
- Fourth column: Rep. Pub. Id
- Fifth column: Gene Title and other information (E.g., Chromosomal Location)

*Note*:

The first five columns may have missing values. Missing or even inaccurate values on the first six columns do not affect our analysis. Note, however, that missing or inaccurate values may cause inaccurate analyses by external resources such as NCBI, GeneMANIA, DAVID, and GCAT…

- Sixth column (and the rest) specifies the expression value of each replicate for a treatment.
- Replicates for each treatment group must be consecutive.
- Expression values must be non-log values.

Figure 1C shows a snapshot of a sample dataset (shown in Excel, but you must convert to text format before uploading).

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Probe Set | Gene Symbol | UniGene ID | Rep. Pub. ID | Gene Title | Chro. Loc | CON1 | CON2 | CON3 | CON4 | CON5 | BNF1 | BNF2 |
| 2 | 1369943_at | Tgm2 | Rn.10 | NM_019386 | transglutaminase 2, C polypeptide | chr3q42 | 2177.697 | 1045.617 | 1941.678 | 1235.958 | 1691.028 | 1032.806 | 1688.829 |
| 3 | 1367657_at | Btg1 | Rn.1000 | NM_017258 | B-cell translocation gene 1, anti-pr | chr7q13 | 8434.484 | 9747 | 7945.055 | 9150.293 | 11070.69 | 7135.597 | 11664.8 |
| 4 | 1368270_at | Apobec1 | Rn.10002 | NM_012907 | apolipoprotein B editing complex 1 | chr4q42 | 19.79208 | 19.51061 | 19.76153 | 14.28283 | 14.48626 | 18.66085 | 20.0334 |
| 5 | 1372094_at | Supt5h | Rn.100021 | BG380556 | suppressor of Ty 5 homolog (S. ce | chr1q21 | 996.1499 | 867.4282 | 1035.416 | 849.5776 | 1071.944 | 1095.092 | 985.2087 |
| 6 | 1399029_at | Usp48 | Rn.100022 | BI285965 | ubiquitin specific protease 48 | chr5q36 | 102.4236 | 118.1206 | 77.58885 | 107.6658 | 103.2081 | 130.8167 | 87.8831 |
| 7 | 1376849_at | Usp48 | Rn.100022 | BM384872 | ubiquitin specific protease 48 | chr5q36 | 217.9693 | 168.7466 | 149.8477 | 170.0002 | 241.9848 | 148.2196 | 145.8392 |
| 8 | 1397005_at | Podxl2_predict | Rn.100042 | BF393900 | Podocalyxin-like 2 (predicted) | chr4q34 | 6.006983 | 6.366794 | 6.387213 | 6.543448 | 6.351457 | 7.211494 | 6.320073 |
| 9 | 1368627_at | Rgn | Rn.10006 | NM_031546 | regucalcin | chrXq11-q12 | 11479.95 | 7131.535 | 6869.644 | 8908.691 | 9825.559 | 8259.493 | 6571.767 |
| 10 | 1388858_at | Map2k3 | Rn.100064 | BI283843 | mitogen activated protein kinase k | chr10q22 | 1239.593 | 1518.014 | 1590.37 | 1718.909 | 2218.131 | 1596.66 | 1284.47 |
| 11 | 1398267_at | Slc22a7 | Rn.10009 /// | NM_053537 | solute carrier family 22 (organic an | chr9q12 | 1029.831 | 712.9044 | 906.3523 | 556.0298 | 627.1698 | 467.5999 | 613.8968 |
| 12 | 1385837_at | Hoxd3_mappe | Rn.100101 | AA956624 | homeo box D3 (mapped) | chr3q23 | 7.942043 | 7.99069 | 7.999732 | 8.325685 | 7.906064 | 8.103678 | 7.995693 |
| 13 | 1371584_at | Trpc4ap | Rn.100109 | BM390843 | transient receptor potential cation | chr3q42 | 419.7459 | 388.7067 | 398.329 | 357.8584 | 360.7175 | 350.181 | 378.3355 |
| 14 | 1391946_at | Selp | Rn.10012 | BI296054 | selectin, platelet | chr13q22 | 8.03969 | 8.102237 | 8.26353 | 8.209499 | 8.096546 | 8.205031 | 8.064649 |
| 15 | 1369813_at | Dnajc5 | Rn.100120 | U39320 | DnaJ (Hsp40) homolog, subfamily | chr3q43 | 57.42489 | 76.12687 | 59.24449 | 52.7556 | 58.62772 | 61.10849 | 86.04957 |
| 16 | 1371870_at | Anxa13_predic | Rn.100125 | AI169493 | Annexin A13 (predicted) | chr7q33 | 1011.147 | 804.3979 | 830.6408 | 755.7788 | 881.1392 | 834.6915 | 726.6166 |
| 17 | 1391346_at | Anxa13_predic | Rn.100125 /// | BG373537 | Annexin A13 (predicted) /// Zinc-fir | chr7q33 | 563.6989 | 521.5713 | 570.9722 | 503.0838 | 625.7846 | 483.1054 | 615.4033 |
| 18 | 1370024_at | RGD1307234_ | Rn.10014 /// | NM_030832 | Similar to RIKEN cDNA 4931400A | chr9q38 /// ch | 5966.042 | 6127.185 | 9237.181 | 4971.16 | 5326.926 | 10443.09 | 7361.715 |
| 19 | 1368600_at | Slc26a1 | Rn.10016 | NM_022287 | solute carrier family 26 (sulfate tra | chr14p22 | 3435.934 | 3397.572 | 3737.338 | 3482.194 | 4325.109 | 3459.891 | 2484.986 |

*Figure* 1C. A snapshot of a dataset in Excel format

**Requests**

a. Choose a dataset file for uploading (with the format as above)

b. Choose significance level p-value for the analysis (default value is 0.05).

c. Choose number of replicates for all treatments of the dataset, values are distinguished by a gap (for example: 5 5 5 5).

d. Enter description of the request if needed (more about dataset, parameters or others) and submit the request.

   See Figure 2C and 3C for more details.

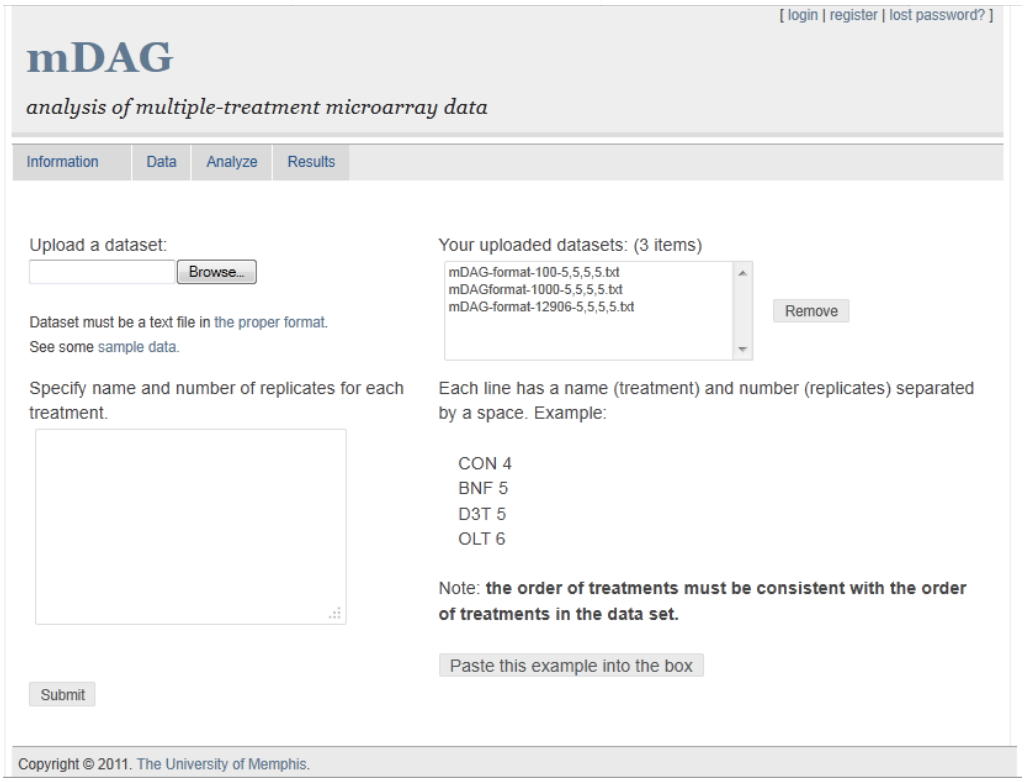


*Figure* 2C: Interface for uploading dataset of mDAG

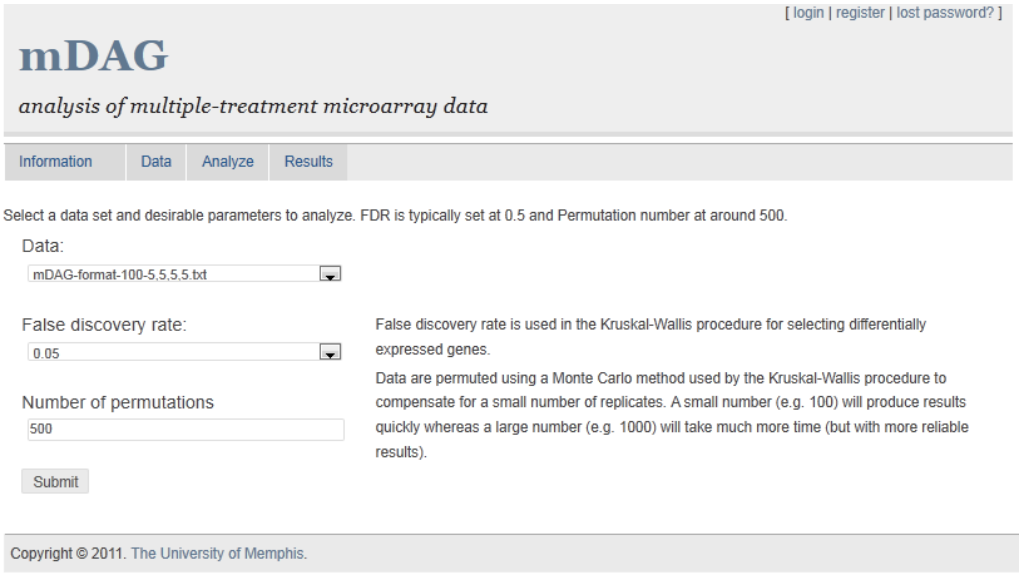


*Figure* 3C: Interface for uploading dataset of mDAG

**Results**

a. User can check information about requests and results, including request description, status of the request and request timing.

   i. Request description includes all information about datasets and their parameters. User can download the dataset files.

   ii. Status of the request includes: Waiting, Running, Completed, and Error. If status is “Completed”, user can see the analyzing result.

   iii. Result timing includes: request upload time, analyzing start time and finish time.

b. Use can rerun the dataset (for both completed and error status).

c. User can remove the request:

   i. With "Waiting" or "Error" status, the system removes the corresponding dataset.

   ii. With "Running" status, the system removes the corresponding dataset and stops the corresponding analysis process.

   iii. With "Completed" status, the system removes both the corresponding dataset and analysis results.

   See Figure 4C for more details.

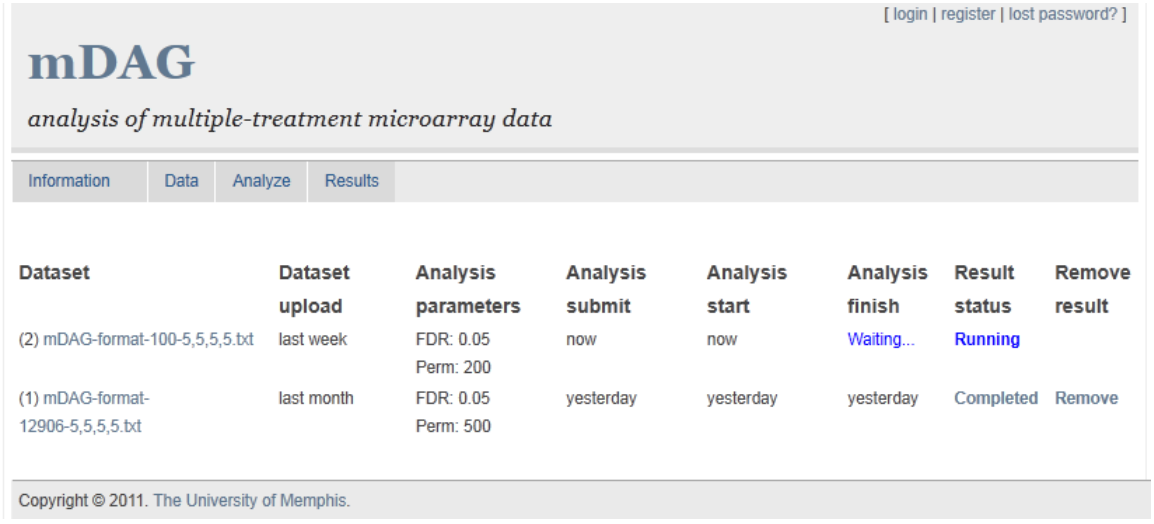

*Figure* 4C: Interface for checking results of mDAG

**Other functions**

a. Manage Database (only admins):
    i. Manage users (add, remove, modify...).
    ii. Manipulate on databases (upload/download/select/insert/delete...).
b. Manage profile:
    i. User can change their profile, password and related information.
    ii. There are two kinds of user: admin and registered user. Admins can use all functions of the website. Registered user can use all functions except administration menu.
    iii. Data belongs to each user and only this user can see or manipulate on their data.

## Appendix D. Main algorithms of mDAG

**Algorithm 1: Scheduler**

*Input:*

dataset_list: list of datasets with related status. Read from Datasets table.

max_process: max number of processes that can be run. Read from Systems table.

*Ouput:*

result_set: set of results

*Begin*

```
LOCK  #Set lock flag
#Count the number of running and waiting processes
running_process=0, waiting_process=0
read dataset_list:
for all dataset in dataset_list:
      if  (dataset.status==Running): running_process++
     elif (dataset.status==Waiting): waiting_process++
#If waiting number = 0 then unlock and halt
if (waiting_process == 0):
      UNLOCK   ## Set unlock flag in Systems table
      return 0
#If running number >= allowed number and waiting number > 0 then wait and call
Scheduler again
elif (running_process >= max_process):
      UNLOCK    ## Set unlock flag
```

wait(x minutes)

Call Scheduler  ##call Scheduler again

#If running number < allowed number and waiting number > 0 then perform and halt

else:

remain_process=0

for all dataset in dataset_list:

if (dataset.status==Waiting):

set dataset.status=Running

Executer(dataset) <put into background>

remain_process++

if (remain_process > max_process - running_process):

break  #break out of for loop

UNLOCK  #Set unlock flag

return 0

*End*

**Algorithm 2:  Executer**

*Input:*

- dataset: input dataset with related status

*Ouput:*

- result: analyzing result

*Begin*

Call MPC program #The analyzing program with results are tDAGs.

##If result exist

if done:

set dataset.status=Completed

return result

##If there is error

elif failed:

set dataset.status=Error

return 0

Call Scheduler ##call Scheduler again

*End*