USING FORMAL CONCEPT ANALYSIS FOR MICROARRAY DATA COMPARISON

V. CHOI *

Department of Computer Science,
Virginia Tech,
660 McBryde Hall,
Blacksburg, VA 24061, USA
E-mail: vchoi@cs.vt.edu

Y. HUANG

Department of Computer Science,
Rutgers University,
110 Frelinghuysen Road,
Piscataway, NJ 08854, USA

V. LAM, D. POTTER, R. LAUBENBACHER, K. DUCA

Virginia Bioinformatics Institute,
Washington Street, MC 0477
Virginia Tech, Blacksburg, VA 24061
Blacksburg, VA 24060, USA

Microarray technologies, which can measure tens of thousands of gene expression values simultaneously in a single experiment, have become a common research method for biomedical researchers. Computational tools to analyze microarray data for biological discovery are needed. In this paper, we investigate the feasibility of using Formal Concept Analysis (FCA) as a tool for microarray data analysis. The method of FCA builds a (concept) lattice from the experimental data together with additional biological information. For microarray data, each vertex of the lattice corresponds to a subset of genes that are grouped together according to their expression values and some biological information related to gene function. The lattice structure of these gene sets might reflect biological relationships in the dataset. Similarities and differences between experiments can then be investigated by comparing their corresponding lattices according to various graph measures. We apply our method to microarray data derived from influenza infected mouse lung tissue and healthy controls. Our preliminary results show the promise of our method as a tool for microarray data analysis.

1. Introduction

Microarray technologies, which can measure tens of thousands of gene expression values simultaneously in a single experiment, across different conditions and over time, have been

*Corresponding author

widely used in biomedical research. They have found many applications, such as classification of tumors, assigning functions to previously unannotated genes, grouping genes into functional pathways etc (see [16] for a review). A large collection of database is available in the public domain (e.g. see [8, 15]). A wealth of methods [13, 5] has been proposed for analyzing these datasets to gain biological insights. A main method for analyzing these microarray data is based on clustering, which groups set of genes, and/or groups of experimental conditions, that exhibit similar expression patterns. These include single clustering algorithms, such as hierarchical clustering, k-means, self-organizing map (SOM) algorithms (see [10] for a review and references therein); and biclustering algorithms (see [12] and references therein). However, the challenge to derive useful knowledge from microarray data still remains. For example, see [3] for a recent biclustering algorithm that is based on non-smooth non-negative matrix factorization. In this paper, we propose another method that is based on Formal Concept Analysis (FCA) [18, 6] as an alternative to the clustering approach.

Our Approach. The method of FCA builds a (concept) lattice from the experimental data together with additional biological information. Each vertex of the lattice corresponds to a subset of genes that are grouped together according to their expression values and some biological information related to gene function. See Section 2 for the background of FCA. The lattice structure of these gene sets might reflect biological relationships in the dataset. Similarities and differences between experiments can then be investigated by comparing their corresponding lattices according to various graph measures. In the high level description, our method consists of the following three main steps:

(1) Build a binary relation (cross-table) for each experiment. The objects of the binary relation are genes; and there are two types of attributes: gene expression attributes and biological attributes. The gene expression attributes are obtained by a *discretization* procedure on gene expression values. The biological attributes can be any biological properties related to gene function.
(2) Construct a Galois/concept lattice for each experiment's binary relation using the efficient Galois/concept lattice algorithm described in [4].
(3) Define a distance measure and compare the lattices.

Note that the biological attributes of genes are invariant/constant for all experiments and they can be preprocessed. The ability to integrate these constant biological attributes is one of the advantages of our method over clustering methods. This is because the constant information will be canceled out in clustering methods and thus do not add any contributions.

Related work of using FCA for microarray data mining. Using FCA for microarray data comparison was proposed in D. Potter's thesis [14]. Based on the framework proposed there, we more rigorously develop each step. In particular, we more carefully discretize the gene expression values to suit our purposes, namely, close gene expression values should share the same attribute; and distance measure is also more rigorously defined and better results are obtained. Also, our concept lattice construction algorithm is very efficient

(within 1 second) while our data sets were too large for the program in[14] to handle. We should also mention that using FCA or Concept lattice approach to mine microarray data were also studied in [1, 2]. The goal there was to extract local patterns in the microarray data and no biological attributes were employed.

Outline. The paper is organized as follows. In Section 2, we review some background and notation on FCA. In Section 3, we describe our method in details. In Section 4, we describe our data and present our preliminary results applying our method to the data. We conclude with future work in Section 5.

2. Background on FCA

Formal Concept Analysis (FCA) [6] is a method that is based on lattice theory for the analysis of binary relational data. It was introduced by Rudolf Wille in 1980s. Since its introduction, FCA has found many applications in data mining, knowledge discovery and machine learning etc [18].

The input of FCA consists of a triple $(\mathcal{O}, \mathcal{M}, \mathcal{I})$, called *context*, where $\mathcal{O} = \{g_1, g_2, \ldots, g_n\}$ is a set of n elements, called *objects*; $\mathcal{M} = \{1, 2, \ldots, m\}$ is a set of m elements, called *attributes*; and $\mathcal{I} \subseteq \mathcal{O} \times \mathcal{M}$ is a binary relation. The context is often represented by a *cross-table* as shown in Figure 1. A set $X \subseteq \mathcal{O}$ is called an *object set*, and a set $J \subseteq \mathcal{M}$ is called an *attribute set*. Following the convention, we write an object set $\{a, c, e\}$ as ace, and an attribute set $\{1, 3, 4\}$ as 134.

For $i \in \mathcal{M}$, denote the adjacency list of i by $\mathsf{nbr}(i) = \{g \in \mathcal{O} : (g, i) \in \mathcal{I}\}$. Similarly, for $g \in \mathcal{O}$, denote the adjacency list of g by $\mathsf{nbr}(g) = \{i \in \mathcal{M} : (g, i) \in \mathcal{I}\}$.

Definition 2.1. The function $\mathsf{attr} : 2^{\mathcal{O}} \longrightarrow 2^{\mathcal{M}}$ maps a set of objects to their common attributes: $\mathsf{attr}(X) = \cap_{g \in X} \mathsf{nbr}(g)$, for $X \subseteq \mathcal{O}$. The function $\mathsf{obj} : 2^{\mathcal{M}} \longrightarrow 2^{\mathcal{O}}$ maps a set of attributes to their common objects: $\mathsf{obj}(J) = \cap_{j \in J} \mathsf{nbr}(j)$, for $J \subseteq \mathcal{M}$.

It is easy to check that for $X \subseteq \mathcal{O}$, $X \subseteq \mathsf{obj}(\mathsf{attr}(X))$, and for $J \subseteq \mathcal{M}$, $J \subseteq \mathsf{attr}(\mathsf{obj}(J))$.

Definition 2.2. An object set $X \subseteq \mathcal{O}$ is *closed* if $X = \mathsf{obj}(\mathsf{attr}(X))$. An attribute set $J \subseteq \mathcal{M}$ is closed if $J = \mathsf{attr}(\mathsf{obj}(J))$.

The composition of obj and attr induces a *Galois connection* between $2^{\mathcal{O}}$ and $2^{\mathcal{M}}$. Readers are referred to [6] for properties of the Galois connection.

Definition 2.3. A pair $C = (A, B)$, with $A \subseteq \mathcal{O}$ and $B \subseteq \mathcal{M}$, is called a *concept* if $A = \mathsf{obj}(B)$ and $B = \mathsf{attr}(A)$.

For a concept $C = (A, B)$, by definition, both A and B are closed. The set of all concepts of the context $(\mathcal{O}, \mathcal{M}, \mathcal{I})$ is denoted by $\mathcal{B}(\mathcal{O}, \mathcal{M}, \mathcal{I})$ or simply $\mathcal{B}$ when the context is understood.

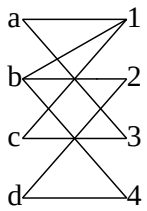
Let (A_1, B_1) and (A_2, B_2) be two concepts in $\mathcal{B}$. Observe that if $A_1 \subseteq A_2$, then $B_2 \subseteq B_1$. We order the concepts in $\mathcal{B}$ by the following relation $\prec$:

$$(A_1, B_1) \prec (A_2, B_2) \Longleftrightarrow A_1 \subseteq A_2 \text{ (and } B_2 \subseteq B_1).$$

It is not difficult to see that the relation $\prec$ is a partial order on $\mathcal{B}$. In fact, $\mathcal{L} =< \mathcal{B}, \prec >$ is a complete lattice and it is known as the *concept* or *Galois* lattice of the context $(\mathcal{O}, \mathcal{M}, \mathcal{I})$. For $C, D \in \mathcal{B}$ with $C \prec D$, if for all $E \in \mathcal{B}$ such that $C \prec E \prec D$ implies that $E = C$ or $E = D$, then C is called the *successor* [a](or *lower neighbor*) of D, and D is called the *predecessor* (or *upper neighbor*) of C . The diagram representing an ordered set (where only successors/predecessors are connected by edges) is called a *Hasse diagram* (or a line diagram). See Figure 1 for an example of the line diagram of a Galois lattice.

When the binary relation is represented as a bipartite graph (see Figure 1), each concept corresponds to a maximal bipartite clique (or maximal biclique). There is also a one-one correspondence between a closed itemset [17] studied in data mining and a concept in FCA. The one-one correspondence of all these terminologies – concepts in FCA, maximal bipartite cliques in theoretical computer science, and closed itemsets in data mining – was known, e.g. [17]. There is extensive work of the related problems in these three communities, see [4] for related literature. The current fastest algorithm given in [4] takes $O(\sum_{a \in \mathsf{ext}(C)} |\mathsf{cnbr}(a)|)$ polynomial delay for each concept C, where $\mathsf{cnbr}(a)$ is the *reduced* adjacency list of a. Readers are referred to [4] for the details.

	1	2	3	4
a	x		x	
b	x	x		x
c	x		x	
d		x		x



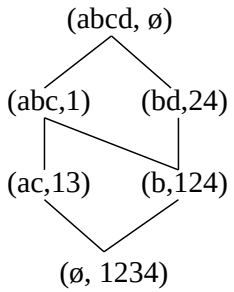


Figure 1. Left, a context $(\mathcal{O}, \mathcal{M}, \mathcal{I})$ with $\mathcal{O} = \{a, b, c, d\}$ and $\mathcal{M} = \{1, 2, 3, 4\}$. The cross $\times$ indicates a pair in the relation $\mathcal{I}$. Middle, the bipartite graph corresponding to the context. Right, the corresponding Galois/concept lattice.

3. Methods

3.1. *Building Binary Relations*

In this section, we describe how to the construct the *context* $(\mathcal{O}, \mathcal{M}, \mathcal{I})$ for each experiment. Here the object set $\mathcal{O}$ consists of a set of genes. There are two types of attributes in the

[a]Some authors called this as immediate successor.

attribute set $\mathcal{M}$: biological attributes and gene expression attributes. For a gene $g \in \mathcal{O}$ and an attribute $a \in \mathcal{M}$, $(g, a) \in \mathcal{I}$ if g has the attribute a.

3.1.1. *Biological Attributes*

Any gene function related properties can be used as biological attributes. For instance, one can use protein motif families as such attributes: a gene has the attribute if its corresponding protein belongs to the motif family. As an example, the protein motif family *oxidoreductases* can be such an attribute, and a gene has this attribute if it is one of the oxidoreductases. There are many other possible biological attributes, such as functional characteristics of a gene, chromosomal location of a gene, known association with disease states etc.

3.1.2. *Discretization of Gene Expression Values*

The data obtained from a microarray experiment consists of a set of genes and each gene has a *gene expression value*. The gene expression values are continuous real numbers. In order to represent microarray data in a binary relation, it requires to discretize the continuous gene expression values into a finite set of values that correspond to attributes. Intuitively, we would like to discretize the gene expression values such that two close values share the same attribute. The straightforward method will be dividing the gene expression values according to large "gaps". First, we sort all the expression values in the increasing order. Let the ordered values be $y_1 < y_2 < \ldots < y_m$. Then we compute the gaps $\delta_i = y_{i+1} - y_i$, for $i = 1, \ldots, m-1$. The idea then is to divide the gene expression values into t subintervals according to the largest $t-1$ gaps. However, the empirical results showed that majorities of these gene expression values were very close. For example, if we partitioned the values according to the largest 4 gaps, more than 75% of these values would belong to one subinterval. If we were to recursively partition this large subinterval, then again, a large portion of the values concentrated on one big subinterval. Instead, after partition the gene expression values into t subintervals, we partition the largest subinterval into s even subintervals.

Recall that our idea is to discretize the gene expression values such that close gene expression values share the same attribute (or a subinterval). However, our even partition might not achieve this goal, for example, two close genes might belong to the same subinterval in one experiment but belong to two consecutive subintervals in another experiment. To overcome this problem, instead of assigning the gene expression value to only one subinterval, if the gene expression value from one of the subintervals is within 50% of the neighbor subinterval, we assign both subintervals to this gene expression value.

We illustrate the discretization procedure by an example in Figure 2. In this figure, $t = 5$ and $s = 4$. That is, we first partition the gene expression values into five subintervals $I_1, I_2, \ldots, I_5$ according to the four largest gaps. And then partition the largest subinterval (I_2 in this example) into four even subintervals. There are total of eight subintervals: $I_1, I_{2_1}, I_{2_2}, I_{2_3}, I_{2_4}, I_3, I_4, I_5$, and each subinterval in the order corresponds to one gene

expression attribute a_i, for $i = 1$ to 8. The gene expression value that falls in the subintervals I_1 (I_3, I_4, I_5 respectively) is assigned to its corresponding attribute a_1 (a_6, a_7, a_8 respectively). The gene expression value that falls in I_2 is assigned to one or two consecutive attributes depending the region it falls. If it falls in the subintervals J_{ii+1} (the region overlapping with 50% of two neighboring subintervals) as shown in Figure 2, then it is assigned to two attributes a_i and a_{i+1}. For example, if a gene expression value falls in the subinterval J_{23} then it gets both attributes a_2 and a_3.

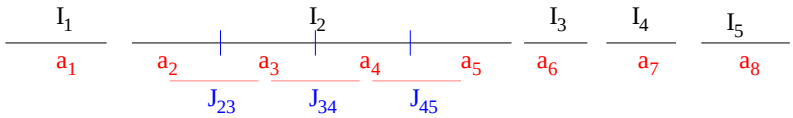


Figure 2. Discretization example. We partition the sorted gene expression values into five subintervals, $I_1, I_2, \ldots, I_5$ according to the four largest gaps. We further partition the largest subinterval (I_2 in this example) into four even subintervals. There are total of eight disjoint subintervals and each subinterval corresponds to one gene expression attribute a_i. The gene expression value that falls in the subintervals I_1 (I_3, I_4, I_5 respectively) is assigned to its corresponding attribute a_1 (a_6, a_7, a_8 respectively). The gene expression value that falls in I_2 is assigned to one or two consecutive attributes depending on the region it falls. If it falls in the subintervals J_{ii+1}, then it is assigned to two attributes a_i and a_{i+1}, for $i = 2, 3, 4$.

3.2. *Concept Lattices Construction*

Once we have the binary relations, we then build a concept lattice for each binary relation, using the efficient algorithm described in [4].

3.3. *Distance Measures for Lattices Comparison*

Given two lattices $L_1 = (V_1, E_1)$ and $L_2 = (V_2, E_2)$, there are many possible distance measures that one can define to measure the similarities or differences of these two lattices. The simplest distance maybe is the one based on common subgraphs. Recall that each vertex in our lattice is a concept and it is labeled by a subset of genes and a subset of attributes. For our purpose, we will ignore all attributes, that is, each vertex is labeled by its object set of genes only. See Figure 3 for an example. A vertex v is called a *common vertex* if v appears in both V_1 and V_2. Let $V_C = V_1 \cap V_2$ be the set of common vertices. For $u, v \in V_C$, if $e = \{u, v\}$ is in both E_1 and E_2. Then e is called a *common edge*. Let E_C be the set of common edges. The distance $\mathsf{distance}(L_1, L_2)$ is then defined by

$$\frac{|L_1 \setminus L_2| + |L_2 \setminus L_1|}{|L_1 \cup L_2|}$$

where $|L_1 \setminus L_2| = |V_1 \setminus V_C| + |E_1 \setminus E_C|$, $|L_2 \setminus L_1| = |V_2 \setminus V_C| + |E_2 \setminus E_C|$ and $|L_1 \cup L_2| = |L_1| + |L_2 \setminus L_1|$ with $|L_1| = |V_1| + |E_1|$.

Since the data is not perfect, we relax our requirement of the definition of a common vertex. Instead of requiring the exact matching of the gene sets of the vertices, we consider

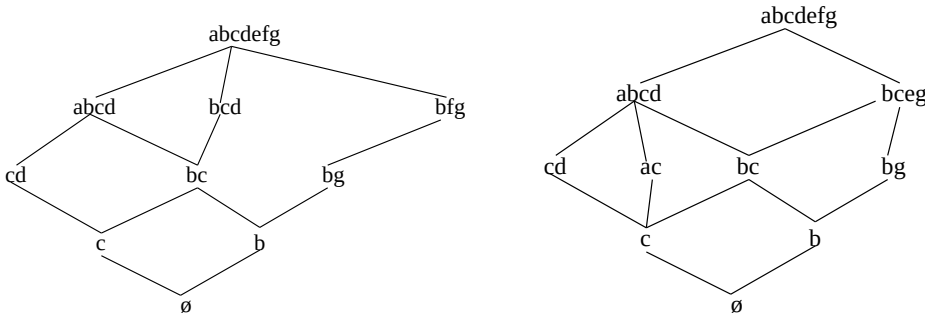


Figure 3. Lattices Comparison. How similar/different are these two lattices?

v_1 and v_2 the same if their gene sets share more than ξ of the maximum size of the two gene sets, i.e., $|\mathsf{obj}(v_1) \cap \mathsf{obj}(v_2)| \geq \xi max(|\mathsf{obj}(v_1)|, |\mathsf{obj}(v_2)|)$.

Many other possible distance measures will be investigated in the future. For example, spectral distance or maximal common sublattice distance that were also mentioned in [14].

4. Our Experiments

4.1. *Our Data*

Our microarray data [11] were derived from the lung tissue of mouse under four different conditions : (1) Control: the mouse was normal and healthy; (2) Flu: the mouse was infected by influenza (H3N1); (3) Smoking: the mouse was forced to smoke for four consecutive days, with nine packs cigarette per day; (4)SmokeFlu: the mouse was both infected with flu and smoking. For each condition, the gene expression values on 6 different time points — 6 hours, 20 hours, 30 hours, 48 hours, 72 hours and 96 hours — were measured. At each time point, there were three replicates, which were used to clean up the noise in the data. Also, the expression values were measured on *probes* and several probes can correspond to a same gene. After cleaning up (through a clean-up procedure written in perl scripts), one gene expression value was obtained for each gene of each sample. There are total of $11,051$ genes for all 24 samples (=6 time points $\times$ 4 conditions).

4.2. *Applying Our Method to the Data*

In this section, we describe the parameters used in our method when it is applied to the above data.

Biological Attributes. We used the protein motif families obtained from PROSITE [9] as our biological attributes. In particular, for our gene sets, we used the stand-alone tools from PROSITE [7] and identified 21 PROSITE families for our gene set. That is, we have 21 biological attributes. Note these biological attributes are experiment independent and thus they only need to be computed once for all experiments.

Discretization of Gene Expression Values. We performed the discretization procedure for the gene expression values (after taking log transformation) of each sample. The parameter t was set to 5, and s was set to 4. That is, there were total of 8 gene expression attributes. We have tested different parameter values and found that increasing the values (and thus number of attributes) did not significantly change the results.

After discretization, we had total of 24 binary relations over the 11051 genes and the 29 attributes. We then applied our lattice construction algorithm on these binary relations to construct the corresponding lattices. There is a small variation of these lattices in terms of its size: each of them has around 530 vertices and 1500 edges. Using the program in [4], it took less than one second to construct each lattice on a Pentium IV 3.0GHz computer with 2.0G memory running under Fedora Core 2 Linux OS.

Distances for Comparing Lattices. The parameter in defining common vertices ξ was set to 70%. That is, $v_1 \in V_1$ and $v_2 \in V_2$, v_1 and v_2 were considered the same if $|\mathsf{obj}(v_1) \cap \mathsf{obj}(v_2)| \geq \xi max(|\mathsf{obj}(v_1)|, |\mathsf{obj}(v_2)|)$, where $\xi = 70\%$. We have tested various relaxation of ξ. This value seems to be best in our current application, that is, it clearly separates different conditions (see Section 4.3 for details).

4.3. *Our Results*

First, using Control sample at 6 hours as a reference, we computed the distances between all other Control samples and all Flu samples to this reference sample. The results are shown in Figure 4. Since the distance measure is not transitive, we also computed all

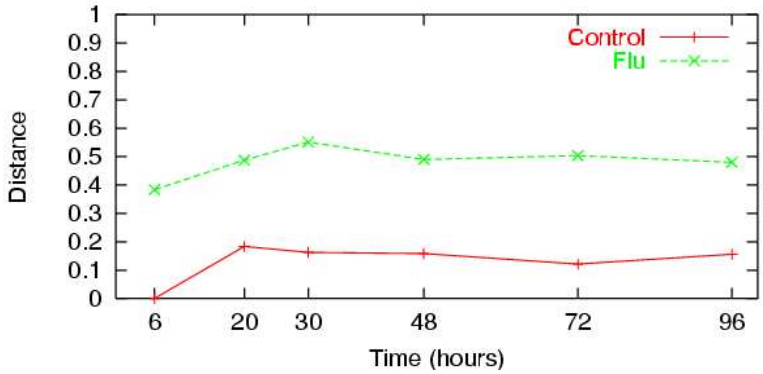


Figure 4. Distance to Control sample at 6 hours (the reference sample). Each data point represents the distance from a sample (Control shown in red, Flu shown in green) to the reference sample. There are small distances from other Control samples to the reference sample. And there is a clear separation between Flu samples from the Control sample at 6 hours.

distances when one of the Control samples was taken as a reference. See Figure 5. The results showed that there was a clear separation between Flu samples from Control samples (regardless which time point of Control was taken as the reference). This is a encouraging

result and we are currently investigating what substructures in the lattices that contribute to the differences which might shed some new biological insights. The results of comparing

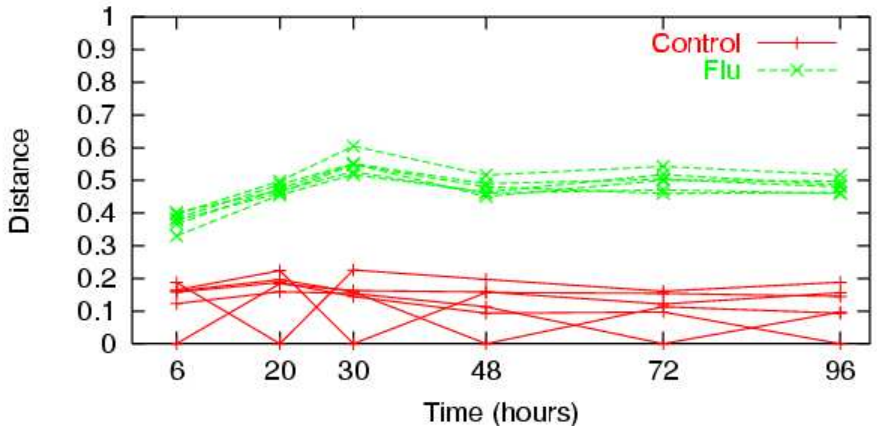


Figure 5. Distance to Control at each time point. There is a clear separation between Flu samples from Control samples regardless which time point of Control is taken as the reference..

other 2 conditions (Smoking/SmokeFlu) is shown in Figure 6.

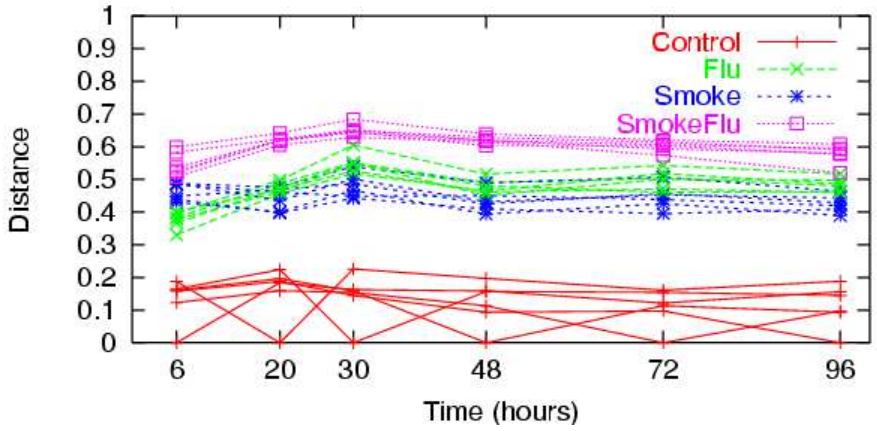


Figure 6. Distance to Control from Flu/Smoking/SmokeFlu.

5. Conclusion and Future Work

Our current preliminary results showed the promise of using FCA approach for microarray data analysis. The distance measure we employed is quite basic and it has not utilized the

properties of the lattice structure. One can investigate other possible distance measures, such as spectral distance, and distance based on maximal common sublattice. These distances will take advantage of the lattice structures and might provide better distinction to aid analyzing the differences between experiments. Beside the global lattice comparison, one can also investigate the local structures of the lattices. These local structures, or sublattices, can be obtained by context decomposition or lattice decomposition. A study of sublattices may assist identification of particular biological pathways or substructures of functional importance.

References

1. J. Besson, C. Robardet, J-F. Boulicaut. Constraint-Based Mining of Formal Concepts in Transactional Data. *PAKDD*, 615–624, 2004.
2. J. Besson, C. Robardet, J-F. Boulicaut, S. Rome. Constraint-based concept mining and its application to microarray data analysis. *Intell. Data Anal.*, 9(1): 59–82, 2005.
3. P. Carmona-Saez, R.D. Pascual-Marqui, F. Tirado, J.M Carazo and A. Pascual-Montano. Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC Bioinformatics*, 7:78, 2006.
4. V. Choi. Faster Algorithms for Constructing a Galois/Concept Lattice. Available at arXiv:cs.DM/0602069.
5. R.C. Deonier, S. Tavare, M.S. Waterman. Computational Genome Analysis: An Introduction. Springer Verlag, 2005.
6. B. Ganter, R. Wille. Formal Concept Analysis: Mathematical Foundations. Springer Verlag, 1996 (Germany version), 1999 (English version).
7. A. Gattiker, E. Gasteiger and A. Bairoch. ScanProsite: a reference implementation of a PROSITE scanning tool *Applied Bioinformatics*, 1:107–108, 2002.
8. L.L. Hsiao, F. Dangond, T. Yoshida, R. Hong, R.V. Jensen, J. Misra, et al. A compendium of gene expression in normal human tissues. *Physiol Genomics*, 7: 97–104, 2001.
9. N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, et al. The PROSITE database. *Nucleic Acids Res.* 34:227–230, 2006.
10. A. Kjersti. Microarray data mining: a survey. SAMBA/02/01. Availalble at http://nr.no/fi les/samba/smbi/microarraysurvey.pdf
11. V. Lam, K. Duca. Mouse gene expression data (Unpublish data).
12. S.C. Madeira and A.L. Oliveira. Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1), 24–45, 2004.
13. G. Piatetsky-Shapiro and P. Tamayo. Microarray Data Mining: Facing the Challenges. *SIGKDD Explorations*, 2003.
14. D.P. Potter. A combinatorial approach to scientifi c exploration of gene expression data: An integrative method using Formal Concept Analysis for the comparative analysis of microarray data. Thesis dissertation, Department of Mathematics, Virginia Tech, August 2005.
15. R. Shyamsundar, Y.H. Kim, J.P. Higgins, K. Montgomery, M. Jorden, et al. A DNA microarray survey of gene expression in normal human tissues. *Genome Biol*, 6:22, 2005.
16. R.B. Stoughton. Applications of DNA Microarrays in Biology. *Annu Rev Biochem.*, 2004.
17. M.J. Zaki, M. Ogihara. Theoretical foundations of association rules. *Proc. 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1–7, 1998.
18. A Formal Concept Analysis Homepage. http://www.upriss.org.uk/fca/fca.html