Gene Ontology Friendly Biclustering of Expression Profiles

Jinze Liu[1], Wei Wang[1], and Jiong Yang[2],
[1]Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599
{liuj, weiwang }@cs.unc.edu
[2]Department of Computer Science, University of Illinois, Urbana-Champaign, IL 61801
jioyang@cs.uiuc.edu

Abstract

The soundness of clustering in the analysis of gene expression profiles and gene function prediction is based on the hypothesis that genes with similar expression profiles may imply strong correlations with their functions in the biological activities. Gene Ontology (GO) has become a well accepted standard in organizing gene function categories. Different gene function categories in GO can have very sophisticated relationships, such as 'part of' and 'overlapping'. Until now, no clustering algorithm can generate gene clusters within which the relationships can naturally reflect those of gene function categories in the GO hierarchy. The failure in resembling the relationships may reduce the confidence of clustering in gene function prediction. In this paper, we present a new clustering technique, Smart Hierarchical Tendency Preserving clustering (SHTP-clustering), based on a bicluster model, Tendency Preserving cluster (TP-Cluster). By directly incorporating Gene Ontology information into the clustering process, the SHTP-clustering algorithm yields a TP-cluster tree within which any subtree can be well mapped to a part of the GO hierarchy. Our experiments on yeast cell cycle data demonstrate that this method is efficient and effective in generating the biological relevant TP-Clusters.

Keywords: Gene Ontology, Gene expression profiles, Biclustering, Tendency Preserving.

1 Introduction

The advent of DNA microarray technologies has revolutionized the experimental study of gene expression. Thousands of genes are routinely probed in a parallel fashion. The gene expression data presents both great opportunities and challenges. They serve as valuable clues to understand the genetic behaviors of life. The complexity of the underlying mechanism underscores the potential complexity in analyzing the gene expression data.

With the advance of microarray technology, the data analysis techniques have been intensively studied as well. Clustering is one of the most popular approaches of analyzing gene expression data without prior knowledge. Several representative algorithmic techniques have been developed and experimented in clustering gene expression data, which include but are not limited to hierarchical clustering [7], self-organizing maps [10], and graphic theoretic approaches, e.g., CLICK [16]. The applicability of clustering in gene function prediction is based on the hypothesis that similar expression profiles imply a functional relation [4] in the biological activities. As a result, the quality of the clusters are often evaluated by their correlations to the known genes' function groups. Those algorithms typically generate a small number of disjoint clusters whose sizes are usually much larger than that of most function categories in GO.

Although these studies have been successful in showing that genes participating in the same biological process have similar expression profiles, there are several reasons preventing the traditional clustering analysis from solving the core issues of modelling biological ontology relationships (Shatkay *et al.*[17]). First, the expression levels of a set of biologically related genes might only show coherency under a subset of conditions. Therefore, clustering over all dimensions (conditions) may separate the biologically related genes from each other. Secondly, grouping genes into disjoint clusters may preclude genes participating in multiple biological activities from being grouped properly [22]. Finally, assume we have a large cluster containing over 200 genes and it includes most of genes from a small function family, e.g., around 10

genes. It is very difficult to tell whether this occurs by chance or not. Therefore, annotating a large gene cluster with a small function family is usually infeasible. Traditional clustering algorithms generating large sizes of clusters only help with the annotation of relatively larger but less specific function categories.

Biclustering (or subspace clustering) might be an answer to solve the above problems. Compared with traditional clustering algorithms, biclustering is capable of discovering the gene expression pattern embedded in only a subset of conditions. In addition, clustering under different sets of conditions generates overlapping biclusters. Cheng and Church [6] are among the pioneers in introducing the concept of biclustering. Their biclusters are based on uniformity criteria, and a greedy algorithm is developed to discover them. Plaid [11] is another model to capture the approximate uniformity in a submatrix in gene expression data and look for patterns where genes differ in their expression levels by a constant vector. Ben-Dor *et al.* [2] discussed approaches for unsupervised identification of patterns in expression data that distinguish two subclasses of a tissue on the basis of a supporting set of genes that offer accurate classification. Tanay *et al.* [20] defined a bicluster as a subset of genes that jointly respond across a subset of conditions. Ben-Dor *et al.* introduced the model of OPSM (order preserving submatrix) [3] to discover a subset of genes identically ordered among a subset of conditions. Probabilistic models were the basis of the work presented above. With probabilistic models, only a limited number of valid clusters may be discovered and a seed usually has to be selected manually before the generation of a cluster. Nevertheless, the inability of revealing the complete set of biclusters hinders the systematic study of the relationships between the biclusters and the function categories in biological activities.

In this paper, we present a biclustering algorithm, Smart Hierarchical Tendency Preserving clustering (SHTP-clustering), which directly incorporates Gene Ontology information into clustering process. A cluster is a Tendency Preserving cluster (TP-Cluster) if it has a maximal subset of genes that have strictly coherent tendency along a subset of conditions. A TP-Cluster is a Smart Tendency Preserving cluster (STP-Cluster) if the enrichments of function categories within the cluster are statistically significant. Our goal is to obtain a tree of STP-Clusters whose hierarchical relationships match the hierarchical organization of the gene function categories in GO. The

clusters may overlap and may vary in size depending on their levels in the STP-cluster tree. We design an algorithm, which automatically constructs a Hierarchical Tendency Preserving clustering tree (TP-cluster tree) in a very compact fashion. We evaluate the mapping from the hierarchical GO relationships to that of the STP-Clusters. The assessment, in turn, is used to guide the mining of STP-Clusters. Our experiments demonstrate that, by directly incorporating gene ontology information into the clustering process, we are able to efficiently and effectively discover biologically relevant TP-Clusters.

The remainder of the paper is organized as follows. Section 2 introduces some preliminary knowledge on GO. Section 3 defines the TP-Cluster and its GO annotation respectively. Section 4 presents the SHTP-clustering algorithm in detail. An extensive performance study is reported in Section 5. Section 6 concludes the paper and discusses some future work.

2 Preliminaries

The GO Consortium was formed to integrate the efforts to regulate the vocabulary for various genomic databases of diverse species in such a way that it can show the essential features shared by all the organisms [23]. GO has three ontology files corresponding to its three categories, namely molecular function, biological process and cellular component. An acyclic directed graph can be obtained for each category with GO terms as nodes. Figure 1 presents a screen shot of the top levels of the gene ontology. At the first level, known genes are classified into three categories, i.e., Molecular Function (MF), Cellular Component (CC) and Biological Process (BP).
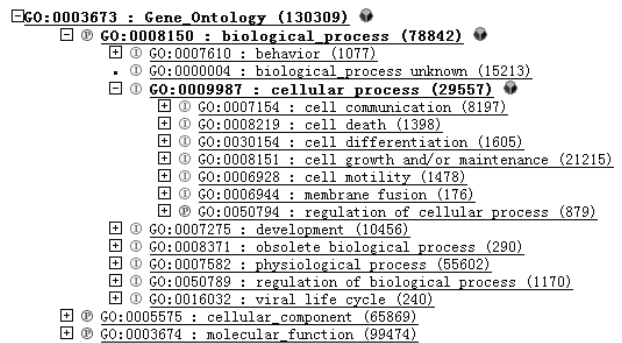

```
⊟GO:0003673 : Gene_Ontology (130309) 🌐
   ⊟ ⑨ GO:0008150 : biological_process (78842) 🌐
      ⊞ ① GO:0007610 : behavior (1077)
      · ① GO:0000004 : biological_process unknown (15213)
      ⊟ ① GO:0009987 : cellular process (29557) 🌐
         ⊞ ① GO:0007154 : cell communication (8197)
         ⊞ ① GO:0008219 : cell death (1398)
         ⊞ ① GO:0030154 : cell differentiation (1605)
         ⊞ ① GO:0008151 : cell growth and/or maintenance (21215)
         ⊞ ① GO:0006928 : cell motility (1478)
         ⊞ ① GO:0006944 : membrane fusion (176)
         ⊞ ⑨ GO:0050794 : regulation of cellular process (879)
      ⊞ ① GO:0007275 : development (10456)
      ⊞ ① GO:0008371 : obsolete biological process (290)
      ⊞ ① GO:0007582 : physiological process (55602)
      ⊞ ① GO:0050789 : regulation of biological process (1170)
      ⊞ ① GO:0016032 : viral life cycle (240)
   ⊞ ⑨ GO:0005575 : cellular_component (65869)
   ⊞ ⑨ GO:0003674 : molecular_function (99474)
```

Figure 1. Schema of GO annotation terms.

Formally, GO hierarchy is naturally described as a

directed acyclic graph (DAG). $GO = <\mathcal{V}, \mathcal{E}>$, where $\mathcal{V}$ is a set of gene function description (GO terms) and $\mathcal{E}$ is a binary relation on $\mathcal{V}$ such that genes with functions described by v_j are a subset of genes with functions described by v_i, denoted $v_j \preceq v_i$, if and only if there exists a path $(v_i, v_{i+1}, ..., v_{j-1}, v_j)$ such that $(v_{m-1}, v_m) \in \mathcal{E}$ for $m = i + 1, i + 2, ..., j - 1, j$. A term's relationship with its ancestor is also defined "part of" or "specific", which means that the set of genes annotated with a GO term is also a subset of the genes annotated with its ancestor GO term.

Nevertheless, to fit GO into our model, we transform the original directed graph of GO into our desired form, an ordered tree. Note that the same GO term may occur several times in an ontology file. From a biological viewpoint, these occurrences should be considered distinct because the location of the term in the hierarchy (i.e., the path from the root to the term) is far more important than the term itself.

Let $\mathcal{D}$ be the universe of the genes and let $a : \mathcal{D} \rightarrow 2^{|\mathcal{V}|}$ be a function annotating each gene with a set of GO-terms at the most specific level of gene ontology. Given a set of GO terms $\mathcal{G} = v_1, v_2, ..., v_t$, a gene is called a *known gene* if there exists a GO term $v, v \in \mathcal{G}$, such that the set of gene-term pairs $\{(x, v) \mid x \in \mathcal{D}$ and $u \in a(x)$ and $u \preceq v$ and $v \in \mathcal{G}\}$ is not empty. Otherwise, the gene is denoted as an *unknown gene*. Unknown genes are either the genes without annotation or genes with annotations beyond the scope of the given GO term set $\mathcal{G}$.

3 TP-Cluster Model and Ontology Interpretation

Let $\mathcal{D}$ be the universe of the n participating genes in a microarray experiment and let $\mathcal{A}$ be the m conditions under which the gene expression levels are measured. The whole gene expression database can be represented in a data matrix $\mathcal{M}$, where M_{ij} is the expression level of gene i under condition j ($0 < i \leq n$, $0 < j \leq m$).

3.1 Tendency Preserving Cluster Model

We are interested in the TP-Clusters, in which the subset of genes in $\mathcal{D}$ exhibits a coherent tendency on the subset of conditions $\mathcal{T}$ of $\mathcal{A}$.

Definition 3.1 *Let $\mathcal{O}$ be a subset of genes in the database $\mathcal{D}$, $\mathcal{O} \subseteq \mathcal{D}$. Let $\mathcal{T}$ be a subset of conditions, $\mathcal{T} \subseteq \mathcal{A}$. Let $\mathcal{R}: \mathcal{T} \times \mathcal{O} \times 2^{|\mathcal{A}|} \rightarrow \mathcal{I}$ be the function that*

gID	a	b	c	d	sequence
1	4002	284	4108	228	dbac
2	401	281	120	298	cbda
3	401	292	109	238	cdba
4	280	318	37	215	cdab

Table 1. An example dataset.

*assigns the rank of a gene i's condition j to be r, if the expression value of the gene i under condition j is the rth lowest value among that under all the conditions in $\mathcal{T}$. ($\mathcal{O}$, $\mathcal{T}$) forms a **TP-Cluster (Tendency Preserving Cluster)**, if $\forall i, j$ ($i, j \in \mathcal{O}$), $\forall a$ ($a \in \mathcal{T}$), $\mathcal{R}(i, a, \mathcal{T}) = \mathcal{R}(j, a, \mathcal{T})$ and $\forall k$ ($k \in \mathcal{D} - \mathcal{O}$), $\forall l (l \in \mathcal{O})$, $\exists b$ ($b \in \mathcal{T}$), $\mathcal{R}(k, b, \mathcal{T}) \neq \mathcal{R}(l, b, \mathcal{T})$.*

Definition 3.1 first defines the rank function $\mathcal{R}$. Based on the rank function, a TP-Cluster is defined as a subset of genes which have consistent ranks along a subset of conditions. In addition, a TP-Cluster is defined to be maximal in that adding any additional gene in the database will violate the rank coherence within the cluster. This property distinguishes our model from OPSM model which also searches for a subset of genes along identically ordered a subset of condition [3]. An OPSM might not be maximal.

For example, in Table 1, we say that the gene set $\{2, 3\}$ forms a TP-Cluster along the subset of conditions $\{a, c, d\}$, since the ranks of the three conditions for both genes are the same, i.e. (3, 1, 2).

Next, we show that each TP-Cluster can be mapped onto an ordered sequence of condition labels by imposing a consistent order of the conditions, such as monotonically increasing or decreasing.

Definition 3.2 *Given a TP-Cluster $\mathcal{C}$ with gene set $\mathcal{O}$ and condition set $\mathcal{T}$, we call a sequence of conditions S representing $\mathcal{C}$ in a monotonically increasing order, if $S = \pi(\mathcal{T})$, where function π places each condition a in $\mathcal{T}$ at the position $\mathcal{R}(a)$ in sequence S.*

For example, for cluster $\{2, 3\} \times \{a, c, d\}$ in Table 1, the sequence of conditions that represents the monotonically increasing order is cda. For condition c, its rank is 1. Therefore, its position in the sequence is 1.

Definition 3.3 *Given two TP-Clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ with condition sets $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively, we call $\mathcal{C}_1$ is an ancestor of $\mathcal{C}_2$ if $\pi(\mathcal{T}_1)$ is a prefix of sequence $\pi(\mathcal{T}_2)$.*

For example in Table 1, the sequence representing cluster $\mathcal{C}_1 = \{2, 3\} \times \{a, c, d\}$ is cda. The sequence representing cluster $\mathcal{C}_2 = \{2, 3, 4\} \times \{c, d\}$ is cd. Since cd

is a prefix of cda, we call $\mathcal{C}_2$ is an ancestor cluster of $\mathcal{C}_1$.

Based on the mapping from the TP-Clusters to the sequences, we are able to organize the TP-Clusters into a prefix tree. We will introduce an algorithm which builds the TP-cluster tree in a very compact fashion in the Section 4.

The following Lemma illustrates the 'part-of' relationships that occur between two TP-Clusters of which one is an ancestor of the other.

Lemma 3.1 *Let $\mathcal{C}$ and $\mathcal{C}'$ be two TP-Clusters in the database $\mathcal{D}$. Let $\mathcal{O}$ and $\mathcal{O}'$ be the gene set of $\mathcal{C}$ and $\mathcal{C}'$ respectively, if $\mathcal{C}'$ is an ancestor of $\mathcal{C}$, then $\mathcal{O} \subseteq \mathcal{O}'$.*

Proof 3.1 *Since $\mathcal{C}'$ is an ancestor of $\mathcal{C}$, $\pi(\mathcal{T}')$ must be a prefix of $\pi(\mathcal{T})$, based on Definition 3.3. $\forall g(g \in \mathcal{O})$, g supports $\pi(\mathcal{T})$. Thus, g must support any prefix of $\pi(\mathcal{T})$, including $\pi(\mathcal{T}')$. Therefore, $g \in \mathcal{O}'$. Since $\forall g, g \in \mathcal{O}$ implies $g \in \mathcal{O}'$, $\mathcal{O} \subseteq \mathcal{O}'$.*

Obviously, clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ in the previous example follow this property. $\mathcal{C}_2$ is an ancestor of $\mathcal{C}_1$ and $\{2,3,4\} \supseteq \{2,3\}$.

3.2 The TP-cluster tree

The TP-cluster tree is generally analogous to a prefix tree of a predefined set of sequences. However, it is also different because of its unique interpretation of each node and the parent-child relationship. Each node in a TP-cluster tree represents a unique TP-Cluster. The root node corresponds to the null space. The nodes at level m correspond to m dimensional TP-Clusters. The TP-Cluster at a node is related to its immediate parent by being part of the cluster. Each TP-Cluster other than the null root is a 1-dimensional extension of its parent cluster. In order to elucidate the structure of TP-cluster tree, we give a complete TP-Cluster tree of three conditions in Figure 2, where each TP-Cluster is represented by a sequence. This tree is 'complete' since there does not exist another TP-Cluster not included in the tree. The figure contains a three-level tree structure which corresponds to 1-, 2- and 3-dimensional TP-Clusters. Each node u in the TP-cluster tree is represented by the path from the root of the TP-Cluster tree to u. For example, the TP-Cluster with two conditions $\{b,c\}$ ordered increasingly as (bc) will be put at the node $-1bc$. The gene set associated with each node in the TP-cluster tree is omitted in the figure.
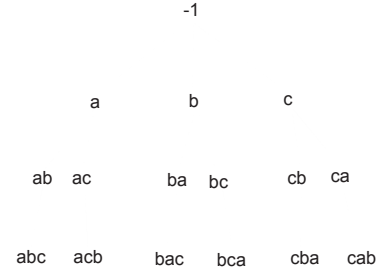

Figure 2. A complete TP-Cluster tree given a condition space $\mathcal{A}=\{a, b, c\}$.

Definition 3.4 *The TP-cluster tree is a hierarchical arrangement of TP-Clusters with the following properties: 1) The tree is rooted at level 0 with -1. 2) Each node at level m corresponds to an m-dimensional TP-Cluster represented by a length-m sequence. 3) Each node at level $(m + 1)$ is a 1-dimensional extension of its immediate ancestor, which corresponds to a length $(m + 1)$ sequence.*

What we are interested in is the hierarchical relationship among a set of TP-Clusters. Investigating the relationships may help us with the prediction of the behavior of higher dimensional clusters based on the lower dimensional ones.

3.3 Annotation of a Gene Cluster

In this subsection, we present the annotation of a gene cluster based on GO. We first introduce the P-value to assess the significance of a particular function group within a cluster.

The hypergeometric distribution is used to model the probability of observing at least k genes from a cluster of n genes by chance in a category containing f genes from a total genome size of g genes. The P-value is given by $P = 1 - \sum_{i=0}^{k} \frac{\binom{f}{i}\binom{g-f}{n-i}}{\binom{g}{n}}$. The test measures whether a cluster is enriched with genes from a particular category to a greater extent than that would be expected by chance. For e

xample, if the majority of genes in a cluster have the same biological function, then it is unlikely that this happens by chance and the category's P-value would be close to 0. Adopting the Bonferroni correction [15] for multiple independent hypotheses, $\frac{0.01}{Na}$ is used as the default threshold θ_p, to measure the significance of the P-value.

To annotate a cluster, the P-value of each category present in the cluster is computed first. Given a cut-off P-value threshold θ_p, categories whose P-value are larger than θ_p are eliminated without further consideration. The result is a set of significant GO function categories $\mathcal{V}=\{v_1, v_2, ..., v_t\}$. There are two naive ways to annotate the clusters with the set of the significant function categories $\mathcal{V}$. One method is to keep all significant function categories as annotation candidates. However, the annotation might become ambiguous if genes in one cluster are assigned with too many function categories. The other way is to annotate a cluster with the category that has the least P-value. Choosing the most significant category to represent the cluster is reasonable. However, the simplicity is at the expense of discarding useful information, such as the distribution of the subcategories of the most significant category.

In our method, we adopt a middle way between the two extremes. We use an appropriate subtree in the GO tree to annotate a cluster. The subtree is rooted at the node of the most significant category and includes all of its significant reachable subcategories. The annotation is formally defined as the Ontology SubTree (OST) in Definition 3.5

Definition 3.5 *Given a cluster $\mathcal{C}$, its significant function categories $\mathcal{V} = \{v_1, v_2, ...v_t\}$, and the directed ontology tree $\mathcal{G}=< \mathcal{V}, \mathcal{E} >$, the Ontology SubTree (OST) $\mathcal{H}=< \mathcal{V}', \mathcal{E}' >$ representing cluster $\mathcal{C}$ is defined as the following: 1.The root of $\mathcal{H}$ is the function category v_r, $0 < r \leq t$, where $P(v_r, \mathcal{C})=\min_{0<i\leq t}(P(v_i, \mathcal{C}))$. 2. $\forall v' \in \mathcal{V}$', there exists a path L ($L \subseteq \mathcal{E}$) leading from v_r to v'. 3. $\forall v_1', v_2' \in \mathcal{V}'$, if $\exists e', e' \in \mathcal{V}$ and e' connects v_1' and v_2', then $e' \in \mathcal{E}'$.*

First, with the hierarchical structure of GO, a gene of a function family will always be a member in the function family's ancestor. Therefore, the OST is rooted at the most significant category. The less significant and less specific ancestor function categories are omitted.

Secondly, although the children of v_r are not as significant as v_r in cluster $\mathcal{C}$, it is still possible that further split of the cluster may signify the coherence of the more specific categories of v. Consequently, an OST representing $\mathcal{C}$ is a maximal connected tree rooted at the most significant category in $\mathcal{C}$.

Figure 3 shows a set of significant function categories of a cluster organized in a tree structure. To determine the OST representing this cluster, we first find
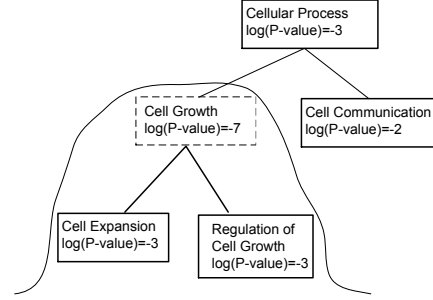

Figure 3. An example of OST representing a Cluster. The categories and P-value are shown at each node. The OST is the subtree under the curve.

out the location of the most significant function group, which in this case is cell growth, with log(P-value)=-7. We then discard its parent category —cellular process, and sibling—cell communication, which have higher P-value. The resulting OST is the subtree rooted at cell growth.

Definition 3.6 *Given a cluster $\mathcal{C}$, we call $\mathcal{C}$ is functionally enriched if there exists an OST representing the cluster, given a P-value threshold θ_p.*

Definition 3.7 defines the $\preceq$ relationship between two OSTs.

Definition 3.7 *Given two OSTs $\mathcal{H}_1$ and $\mathcal{H}_2$, we call $\mathcal{H}_1 \preceq \mathcal{H}_2$ if the root node of $\mathcal{H}_1$ appears as a node of $\mathcal{H}_2$.*

For example, Figure 4 contains two clusters' OSTs. We call $\mathcal{H}_2 \preceq \mathcal{H}_1$ since we can find the root node cellular growth of $\mathcal{H}_2$ in $\mathcal{H}_1$.

3.4 Mapping the TP-cluster tree onto GO Hierarchy

A child-parent relationship in GO hierarchy is "part-of" and "more specific than". Or, in other words, the genes in the child term should be more similar and consistent than those in the parent term. Here we assume that the subset of conditions a child term may stay close on is larger than that for its parent term in the GO hierarchy. This is exactly the child-parent relationship in a TP-cluster tree. By the same child-parent relationship, we unite the two hierarchies together. Next, we use the child-parent relationship of gene ontology to evaluate the child and parent relationship in the TP-cluster tree.
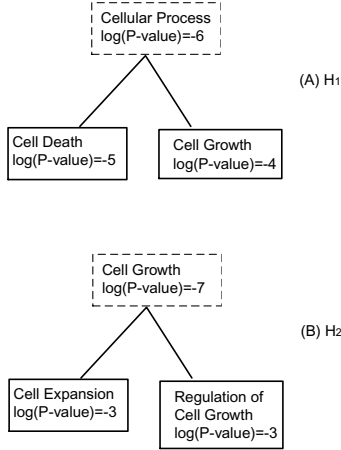
5

Figure 4. An example of two $OSTs$ $\mathcal{H}_1$ and $\mathcal{H}_2$, $\mathcal{H}_2 \preceq \mathcal{H}_1$.

Definition 3.8 *Let $\mathcal{C}$ be a TP-Cluster and $\mathcal{C}'$ be one of $\mathcal{C}$'s descendants. Let $\mathcal{H}$ be $\mathcal{C}$'s OST, and let $\mathcal{H}'$ be $\mathcal{C}'$'s OST. $\mathcal{C}'$ is a biological descendent of $\mathcal{C}$ if $\mathcal{H}' \preceq \mathcal{H}$.*

For instance, $\mathcal{C}_2$ represented by $\mathcal{H}_2$ in Figure 4 is a biological descendant of $\mathcal{C}_1$ represented by $\mathcal{H}_1$.

Problem Statement Let $\mathcal{D}$ be a database with gene set $\mathcal{O}$ and condition set $\mathcal{A}$. Given a threshold θ_p for category enrichment and the GO files, our goal is to extract a biologically relevant hierarchy of enriched TP-Clusters.

Since genes and conditions in the expression data correspond to rows and columns in the expression matrix, we may use the two sets of terms interchangeably in the following sections.

4   Construction of Biologically Relevant TP-cluster tree

In this section, we present the algorithm to build a STP-cluster tree. In order to illustrate the development of a STP-cluster tree, we start from the development of a TP-cluster tree (HTP-clustering). Ontology-based pruning techniques are then added into HTP-clustering process to extract the STP-cluster tree (SHTP-clustering).

4.1   HTP-clustering

The HTP-clustering constructs the TP-cluster tree by suffix concatenation in conjunction with extracting only biologically relevant TP-Clusters. The inputs to the HTP-clustering include the database $\mathcal{D}$, GO, and function enrichment threshold θ_p. The TP-cluster tree is constructed hierarchically in a top-down fashion, along which the dataset $\mathcal{D}$ is partitioned. The HTP-clustering adopts a depth-first pre-order traversal algorithm in order to build the TP-cluster tree. We prefer the depth-first order over the breadth-first order because we can minimize the amount of storage needed for each level to develop clusters at the next level. The depth-first traversal definitely guarantees the correctness of the result because for each node, the construction of its subtree is independent of the construction of its siblings.

The HTP-clustering process can be summarized in two steps:
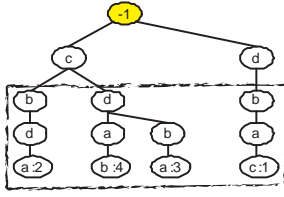
1. We first preprocess the data. Each row in the data matrix will be converted to an ordered sequence of column labels based on the rank function in Definition 3.1. Those sequences will be the inputs to the next step. An initial prefix tree containing the sequence of every gene in the database will be constructed.

2. The ontology information of genes is fed into the TP-cluster tree at the root level. The initial prefix tree is recursively visited and developed in the depth-first order to reveal all frequent subsequences, which represent TP-Clusters. Ontology-based pruning is performed when visiting each node.

We focus on the second step which is more challenging and important during the whole mining process. The data structure representing the TP-cluster tree is defined below.
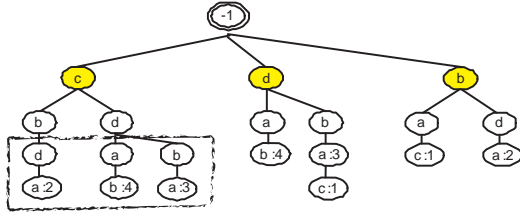1. It consists of one root labelled as "-1" and a set of subtrees as the children of the root;
2. Each node (expect the root) has four entries: entry value, a link to its first child node, a link to its next sibling node, and the list of gene IDs, each of which has a suffix corresponding to the path from the root to this node. In other words, the gene IDs are only recorded at the node that marks the end of a common subsequence.

We use the dataset in Table 1 in the following example to illustrate the suffix concatenation step during the tree construction process.

Example 4.1 *For sequences in Table 1, the initial prefix tree representing the whole database is presented in*

6

(A) Initial tree

(B) First suffix concatenations at level 1

Figure 5. The illustration of suffix tree concatenation.

Figure 5 (A) and the suffix concatenation upon visiting the first node "-1" is illustrated in Figure 5 (B).

Let's denote the node currently being visited as the active node. Given an active node in the TP-cluster tree construction process, for example, at the root "-1" in Figure 5 (B), the suffixes to be inserted to "-1"'s subtree are those inside the rectangle box shown in Figure 5 (A). The concatenation of the suffixes to the current active node is done by merging the suffix tree of the active node with the corresponding subtree one level below the active node. For example, suffix tree "-1cd" in (A) is merged with "-1d". The generated subtree is shown as the "-1d" subtree in (B). (B) is the subsequent tree after the visit of the node "-1". The same procedure will be applied recursively in the depth-first order to construct the TP-cluster tree. For example, after the first node visit at the root "−1", the next node to be visited is "-1c" and the suffixes inside the rectangle box in Figure 5(B) are the next set of suffixes to be inserted. The TP-Cluster algorithm without any biological assessment is presented in Algorithm *growTree*.

Algorithm *growTree*(H,depth)
Input: $\mathcal{H}$: the root of the initial tree,
Output: TP-Cluster existed in $\mathcal{H}$
($*$ Grow patterns on the initial TP-Cluster $\mathcal{H}$ $*$)
1.   **if** $\mathcal{H} = $ **nil**
2.   **return**;
3. $\mathcal{H}_{child} \leftarrow \mathcal{H}$'s first child;
4.   **for** any sub-tree $sub\mathcal{H}$ of $\mathcal{H}$
5.   **do** insertSubTree($sub\mathcal{H}$, $\mathcal{H}$);
6.   growTree($\mathcal{H}_{child}$,$depth + 1$);
7.   growTree($\mathcal{H}$'s next sibling,$depth$);
8.   **return**.
9.

The correctness of the construction of TP-cluster tree is proved in Lemma 4.1.

Lemma 4.1 *Given a database $\mathcal{D}$, the TP-cluster tree contains all TP-Clusters embedded in the database.*

Rationale: According to Definition 3.2, each TP-Cluster corresponds to a unique sequence of the conditions. Therefore, the proof of Lemma 4.1 is equivalent to the proof that the TP-cluster tree contains all frequent subsequences of the set of sequences representing rows in the database. Given any subsequence S', we want to prove that all sequences containing S' will be projected onto the path corresponds to S'. Given any sequence $S = x_1 x_2 x_3 x_4 \ldots x_n$, we want to show that all subsequences of S can be found in a path starting from the root. S is inserted into the tree during the initiation procedure. Then given any subsequence $SS = x_i x_j \ldots x_s, (1 \leq i, s \leq n)$, we can obtain SS by the following steps. First, at node x_i, insert suffix $x_i x_{i+1} \ldots x_n$. Now in the subtree rooted at x_i, node x_j can be found because it should be along the path $x_i x_{i+1} \ldots x_n$ that is inserted in the first step. Similarly, we insert the suffix $x_j \ldots x_n$. As a result, we get the path $x_i x_j x_{j+1} \ldots x_n$. By repeating the same procedure until we insert the suffix starting with x_s, we get the path $x_i x_j \ldots x_s$. Since the path representing a subsequence is unique, all sequences containing S' fall on the node corresponding to S'. The TP-cluster tree contains all the subsequences, or, equivalently, all TP-Clusters embedded in database $\mathcal{D}$.

4.2 SHTP-clustering

Based on the above HTP-clustering, we develop the SHTP-clustering algorithm by incorporating ontology knowledge into the clustering process. The ontology information serves for the following two purposes: (1) the assessment of function enrichments of a cluster. (2) the guidance to select the subset of conditions critical to a function category. (Along that subset of conditions, a significant number of genes in that function category may stay close.) These two functionalities of ontology information are transformed into two pruning techniques in the SHTP-clustering algorithm.

The first pruning technique is based on the distribution of function groups in a cluster. For any cluster $\mathcal{C}$, we expect that there exists at least one function category in $\mathcal{C}$ that is statistically significant. Given a cluster $\mathcal{C}$ and the distribution of categories, we use the following lemma for early detection of the potential appearance of significant function categories.

Lemma 4.2 *Let $\mathcal{C}$ be a cluster, let $\mathcal{V}=\{v_1, v_2, ..., v_t\}$ be a set of function categories and let S be a counter vector in which each element s_i records the number of appearances of the genes in category v_i in $\mathcal{C}$. Let the minimum number of genes required in a cluster be n_r and let θ_p be P-value threshold. $\forall v_i, v_i \in \mathcal{V}$, let $\mathcal{C}_i$' be a cluster with size n_r and contains $\min(s_i, n_r)$ genes in function category v_i. If $\forall\, i,\, P(s_i, \mathcal{C}_i') > \theta_p$, then $\mathcal{C}$ will not become an enriched cluster.*

Proof 4.1 $\forall v_i, v_i \in \mathcal{V}$, *we have $P(s_i, \mathcal{C}_i') \leq P(s_i, \mathcal{C})$ based on the property of P-value, i.e, the P-value increases as the number of genes in the same cluster increases. According to the condition in the Lemma, we have $\theta_p \leq \mathcal{C}_i' \leq P(s_i, \mathcal{C})$. Hence, according to Definition 3.6, $\mathcal{C}$ will not be functionally enriched in any of the function categories in $\mathcal{V}$.*

The algorithm of this pruning technique takes at most $O(|\mathcal{V}|)$ by scanning through each of the function categories in $\mathcal{V}$ and computing its smallest P-value that might occur.

The second technique is to use OST extracted in a parent cluster to guide the selection of its descendent TP-Cluster clusters, by favoring biological children clusters defined in Definition 3.8. Our criterion is based on the hypothesis that, the TP-Clusters in the higher dimensional space are enriched in more specific categories.

Criterion 4.1 *Let $\mathcal{C}$ and $\mathcal{C}'$ be two clusters and let $\mathcal{C}$ be the parent of $\mathcal{C}'$ in the TP-Cluster hierarchy. We say that the development of $\mathcal{C}'$ is not viable if $OST_{\mathcal{C}'} < OST_{\mathcal{C}}$.*

The extraction of OST from a cluster also takes $O|\mathcal{V}|$ if we represent each gene category with a GO code[12]. At most two scans of the ordered GO codes are necessary to generate the OST. Combining the two pruning techniques, we apply the following procedure at each node of the traversal.

1. Evaluate the prediction potential of the cluster corresponding to this node. If it has no potential to become a functionally enriched cluster according to Lemma 4.1, stop further development of this node and its descendants, then go to the next node in the predefined traversal order. Otherwise, go to step 2.

2. Extract the OST of the cluster. If the OST is not biologically viable according to Criterion 4.1, stop further development of this node and its descendants. Go to the next node in the predefined traversal order.

We present the SHTP-clustering algorithm of extracting biologically relevant TP-Clusters in Algorithm *smartGrowTree*. Its major differences from the HTP-clustering algorithm are the recursively feeding and pruning of the OST structure, and the cluster evaluation and pruning based on the significance of the OST.

Algorithm *smartGrowTree*(H, n_c, n_r, depth, parentOST)
Input: $\mathcal{H}$: the root of the initial tree.
Output: TP-Cluster existed in $\mathcal{H}$, the original OST.
($*$ Grow patterns on the initial TP-Cluster $\mathcal{H}$. $*$)
1.　**if** $\mathcal{H} =$ **nil**
2.　　　**return**;
3.　$\mathcal{H}_{child} \leftarrow \mathcal{H}$'s first child;
4.　**for** any sub-tree $sub\mathcal{H}$ of $\mathcal{H}$
5.　　**do** insertSubTree($sub\mathcal{H}, \mathcal{H}$);
6.　$curOST=$ extractOST($\mathcal{H}$);
7.　**if** ($curOST$ is not empty)
8.　　　**if** ($curOST \preceq parentOST$)
9.　　　　　growTree($\mathcal{H}_{child}, n_c, n_r, depth + 1, curOST$);
10.　　　**else** growTree($\mathcal{H}_{sib}, n_c, n_r, depth + 1, parentOST$);
11.　**else**
12.　　　$potential =$ evalFunction($\mathcal{H}, parentOST$)
13.　　　**if** ($potential = good$)
14.　　　　　growTree($\mathcal{H}_{child}, n_c, n_r, depth + 1, curOST$);
15.　　　**else** growTree($\mathcal{H}_{sib}, n_c, n_r, depth, parentOST$);
16.　**return**.

Analysis of HTP-clustering and SHTP-clustering construction For both HTP-clustering and SHTP-clustering, only one scan of the entire data matrix is needed during the clustering. Each row is first converted into a sequence of column labels. The sequences are then inserted into the prefix tree. In the initial tree structure, sequences with the same prefix naturally fall onto the same path from the root to the node corresponding to the end of prefix. To be memory efficient, the row/gene IDs associated with each

path are only recorded at the node marking the end of the longest common prefix shared by these sequences. The depth-first pre-order traversal is then applied to the prefix tree to generate a TP-cluster tree. Pruning techniques based on ontology knowledge further produce a STP-cluster tree without generating the complete TP-cluster tree.

Both the time and space complexities of the two algorithms are exponential determined by the nature of being a NP-hard problem. In the worst case scenario, given a gene expression matrix ($n \times m$), the size of tree is ($\sum_{s=1}^{m} s!\binom{m}{s}$). However, since we use the depth-first traversal of the tree and the part of tree that has been traversed will not be needed for future mining, they can be deleted and the space can be reused. At level $i(i \neq 0)$, we only need to keep $(m - i + 1)$ nodes. Therefore, the maximal space to be allocated during the running time will be limited to $O(n \sum_{i=1}^{m} m - i + 1) = O(n * m^2)$.

The SHTP-clustering will be more space and time efficient than HTP-clustering since less biologically irrelevant TP-Clusters are generated due to the ontology guided pruning techniques. The pruning effects are largely determined by the relationship between a TP-Cluster and the significance of its underlying functional categories.
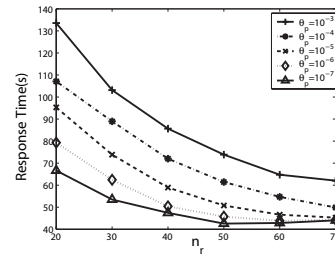
5 Evaluation

Our experiments demonstrate the applicability of SHTP-clustering algorithm in clustering biologically related genes with effective pruning techniques based on GO. The results are evaluated through the comparison of HTP-clustering and SHTP-clustering, and the mapping between the TP-cluster tree to the GO hierarchy. The algorithm was implemented in C and executed on a Linux machine with a 700 MHz CPU and 2G main memory.

Both HTP-clustering and SHTP-clustering algorithms are tested on the yeast cell cycle data of Spellman *et al.* The study monitored the expression levels of 6,218 S. cerevisiae putative gene transcripts (genes) measured at 10-minute intervals over two cell cycles (160 minutes) with 18 time points. Spellman *et al.* identified 799 genes that are cell cycle regulated. We used the expression levels of the 799 genes across 18 time points as the original input matrix. The clustering procedure groups together genes on the basis of their common expression tendency across a subset of time points.
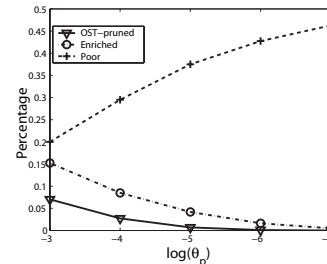
To assess the biological relevance of the clusters, we use GO and P-value to evaluate whether the cluster has significant enrichment in one or more function groups. The ontology of the 799 yeast genes is downloaded from gene ontology consortium [23] in Feb, 2004. We use functions from the three categories: molecular function(MF), cell component(CC) and biological process(BP). We extract categories between ontology level 2 and level 5 with a family size of at least 5. The discovered TP-Clusters in each level of the hierarchy are evaluated for enrichment with any of those function categories.

Types	#Known genes	#Categories ($\#genes > 5$)	#Anno per gene
MF	370	16	0.77
CC	616	48	3.4
BP	538	38	5.72

Table 2. Statistics for the three categories.
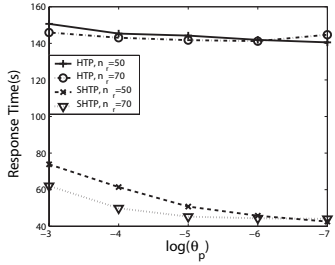

a) Response Time
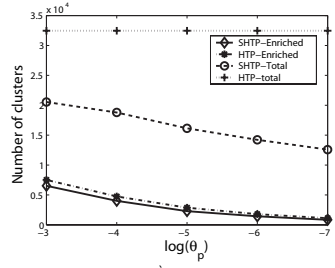

b) The distribution of the clusters

Figure 6. The performance of the SHTP-clustering varying n_r and θ_p.

5.1 Performance

The first set of experiments was done using the SHTP-clustering algorithm and cellular component ontology category to evaluate the performance by varying parameters n_r and θ_p. As shown in Figure 6

9

a) The comparison of response time

b) The comparison of enriched clusters and total clusters

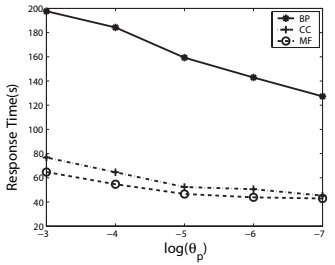**Figure 7. The comparison between SHTP-clustering and HTP-clustering.**

Figure 8. The comparison of performance of SHTP-clustering among three categories.

a), the response time of the $SHTP-clustering$ algorithm decreases as the significance threshold decreases and as the minimum number of rows increases. According to the pruning strategy Lemma 4.1, high significance threshold allows early drop of cluster with poor functional implication. More early pruning enables shorter response time. The n_r helps to prune clusters with the size limitation. The application of the same algorithm to the other two categories exhibits the same trend when varying n_r and θ_p.

Figure 6 b) presents the distribution of the generated clusters in three categories: not enriched cluster, enriched cluster, and enriched cluster not following its parent's OST according to Criterion 4.1. The percentage of not enriched cluster increases significantly as

θ_p decreases. It also explains the performance gain of SHTP-clustering at the same time. Also the percentage of clusters being pruned due to Criterion 4.1 drops significantly compared to the percentage of the enriched clusters as the significance threshold decreases. This may also indicate that the more significant the enrichment of the clusters, the higher the probability that its OST leads to the right direction of selecting the biologically appropriate biclusters.

The second set of experiment in Figure 7 a) is a comparison between SHTP-clustering algorithm and HTP-clustering algorithm. For each algorithm, we have done two tests with different settings of n_r. We can observe significant and consistent improvement of SHTP-clustering algorithm over HTP-clustering especially when θ_p is relatively low. The performance of SHTP-clustering can be as short as 1/4 of that of the HTP-clustering algorithm.

Figure 7 b) compares the number of enriched clusters and the total number of clusters for both of the algorithms HTP-clustering and SHTP-clustering. Clearly, HTP-clustering algorithm generate a large number of TP-Clusters, of which only a small percent are enriched. Compared with HTP-clustering, SHTP-clustering generates less than half of the number of TP-Clusters and almost the same number of enriched TP-Clusters. Overall, SHTP-clustering improves the performance with minimum loss of the enriched clusters.

Figure 8 gives the comparison of the response times varying the three available ontology files, i.e, MF, CC, BF. We can observe a clear trend that the experiments using biological process category consistently spend more time than the rest two. This can be explained by the data in Table 2. The average number of categories that a gene might have is 5.7, which is much higher than that of either the cellular component or the molecular function files. With fewer categories but more gene annotations, the distribution of function groups in a cluster has a higher probability being more concentrated in one or more function groups rather than being evenly distributed. As a result, fewer functional clusters might be pruned, and hence, the response time is longer. In addition, this may also be coincident with the hypothesis that similar gene expression profiles may indicate a function relation in biological process [4]. As a result, more time will be taken for generating a larger number of significantly enriched clusters compared with the rest two ontology files.

Overall, our experiment shows that the ontology-based pruning is effective in reducing the search space of biclustering. In addition, the response time of our algorithm is influenced by the two input parameters and the distribution of genes in each category of the ontology.

5.2 Mapping between GO and the TP-cluster tree

We present a generic example of hierarchically organized clusters that map to a hierarchical substructure of GO.

In Figure 9, (A) presents a three-level hierarchy of TP-Clusters, while (B) shows the corresponding OSTs. The gene ontology summarizing the relationships among all the function categories appearing in (B) is "Necleoside $\rightarrow$ DNA metabolism $\rightarrow$ DNA repair".

The root cluster C_{01} in (A) is the largest cluster with 71 genes. However, it has the smallest number of conditions shared by all genes in its cluster, i.e. $(4, 15, 13, 8)$. Its OST shown at the top of the hierarchy in (B) is rooted at the category, Necleoside. As we go down the hierarchy of clusters in (A), clusters tend to contain a smaller number of genes but share a larger number of consistent conditions. In addition, the OSTs is likely to exist in the subtree of the OST of its parent cluster. For example, the root cluster C_{01} is split into two smaller overlapping clusters C_{11} and C_{12} featuring enriched function "DNA metabolism", which is a subcategory of $necleoside$. $OST_{C_{11}}$ and $OST_{C_{12}}$ suggest that the two clusters in level one have more significant grouping at a deeper level in GO hierarchy than cluster C_{01}. A further clustering of cluster C_{12} into C_{21} with six conditions again signifies a even deeper function group, i.e. "DNA repair".

This example illustrates the connection between the ontology hierarchy and the STP-cluster tree. Our experiments demonstrate that it is possible that only a subset of conditions matter for a ontology category. In addition, the deeper the level of a category within the GO hierarchy, the more the conditions under which the genes in that category have the similar expression profiles.
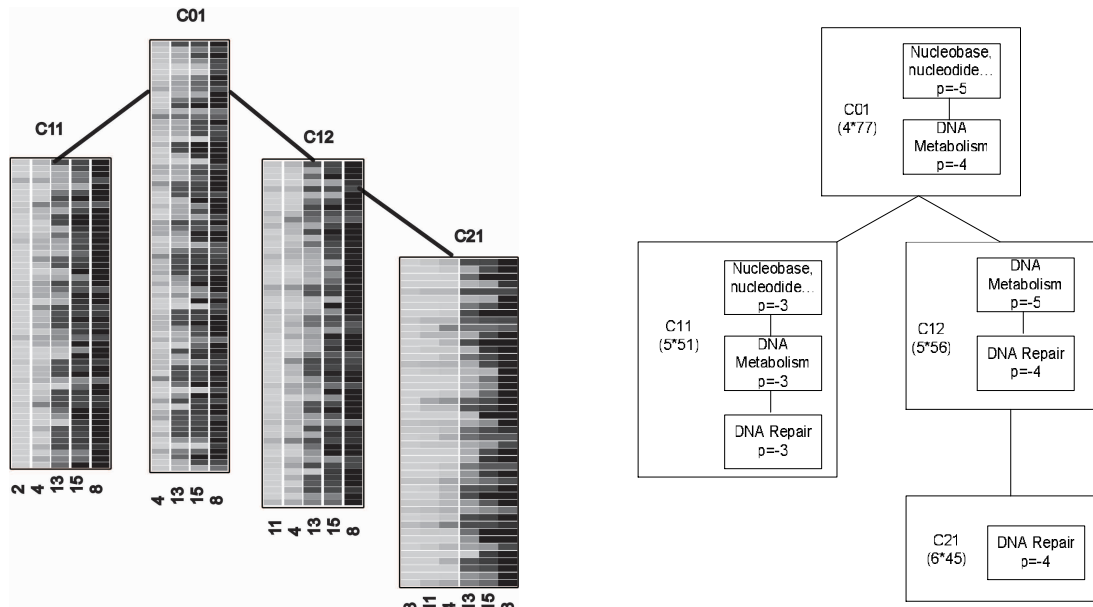
6 Conclusions and Future Work

Clustering of gene expression data has been used for gene function prediction based on the hypothesis that similar expression profiles indicate a function relation in biological activities. However, traditional clustering algorithms are weak in modelling the hierarchy of GO due to the fact that traditional algorithms cannot produce a hierarchy of overlapping clusters with various sizes. To overcome these problems, we propose to use a hierarchy of TP-Clusters to match the hierarchy of GO. We present a biclustering algorithm guided by GO, SHTP-clustering, which efficiently and effectively extracts the biological relevant gene clusters. Our experiments on yeast gene expression data demonstrate the effectiveness of the ontology-based pruning techniques. Our future work will be using the generated STP-cluster tree for effective classification of the unknown genes.

References

[1] A. Ben-Dor, R.Shamir, and Z.Yakhini. Clustering gene expression patterns. in *J Comput Biol* 6(3-4):281-97.

[2] A. Ben-Dor, N. Friedman and Z. Yakhini. Class discovery in gene expression data. In *RECOMB* 2001.

[3] A. Ben-Dor, B. Chor, R.Karp, and Z.Yakhini. Discovering Local Structure in Gene Expression Data: The Order-Preserving Submatrix Problem. In *RECOMB* 2002.

[4] M.P.S. Brown, W.N. Grundy, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machine. *proc. Natl Acad. Sci.* USA, 97, 262-267.

[5] Y. Radmacher M.Bittner M.simon R.Meltzer P.Custerson B.Esteller M.Raffeld M.Yakhini Z.Ben-Dor A.Dougherty E.Kononen J.Bubendorf L.Fehrle W. Pittaluga S.Gruvberger S.Loman N.Johannsson O.Olsson H.Wilfond B.Sauter G.Kallioniemi O.-P.Borg A.Trent J.Hedenfalk I., Duggan D. Gene-expression profiles in hereditary breast cancer. *NEJM*, 344:539-548, 2001.

[6] Y. Cheng and G. Church. Biclustering of expression data. In *ISMB*, 2000.

[7] M.B.Eisen, P.T.Spellman, P.O.Brown, and D.Botstein. Cluster analysis and display of genome-wide expression patterns. In *Proc Natl Acad Sci U S A*, 95(25):14863-8, 1998.

[8] T. R. Hvidsten, A. Lagreid and J. Komorowski. Learning rule-based models of biological process from gene expression time profiles using Gene Ontology. *Bioinformatics*, 2003.

[9] G. Getz, E. Levine and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. Natl Acad. Sci.* USA, 97, 12079-12084, 2000.

[10] S. Kaski, J. Nikkil, G. Wong, Analysis And Visualization Of Gene Expression Data Using Self-Organizing Maps, *Proceedings of NSIP-01, IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing, 2001.*

(A) Expression profiles of the TP-Cluster subtrees   (B) Corresponding OSTs of clusters in (A)

Figure 9. An example of mapping from a hierarchy of TP-Clusters to their OSTs. For each cluster in (A), the rows correspond to the genes while the columns correspond to the conditions.

[11] L.Lazzeroni and A.Owen. Plaid models for gene expression data. 2000

[12] S.G.Lee, J.U.Hur, and Y.S.Kim. A graphic-theoretic modeling on GO space for biological interpretation of gene clusters. *Bioinformatics*, 2004

[13] J.Liu and W.Wang. Subspace clustering by tendency in high dimensional space. In *ICDM 2003*.

[14] E. Segal, B. Taskar, A. Gasch, N. Friedman and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17, S243-S252, 2001.

[15] J. P. Shaffer, Multiple Hypothesis Testing. *Ann. Rev. Psych*. 46, 561-584, 1995

[16] R. Sharan and R.Shamir. Click: A clustering algorithm with applications to gene expression analysis. In *ISMB*, pages 307-216, 2000

[17] H. Shatkay, S. Edwards, W. Wilbur, and M. Boguski Genes, themes and microarrays-using information retrieval for large-scale gene analysis. In *ISMB*, 2000, pp 317-328.

[18] G. Sherlock. Analysis of large-scale gene expression data. *Curr. Opin. Immunol.*, 12, 201-205

[19] P.T. Spellman, G.Sherlock, M.Q.Zhang, V.R.Lyer, K.Anders, M.B.Eisen, P.O.Brown, D.Botstein, and Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast sacccharomyces cerevisiae by microaray hybidization. *Molecular Biology of the Cell*, 9:3273-2297, 1998.

[20] A.Tanay, R. Sharan and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, Vol.18, pages S136-S144, 2002.

[21] S. Tavazoie, J. Hughes, M.Campbell, R. Cho, and G. Church. Yeast micro data set. In *http://arep.med.harvard.edu/biclustering/yeast.matrix*, 2000.

[22] L.F.Wu, T.R.Hughes, A.P.Davierwala, M.D.Robinson, R.Stoughton, S.J.Altschuler: Large-scale prediction of Saccharomyces cerevisiae gene function using overlapping transcriptional clusters.*Nature Genetics* 2002, 31:255-265

[23] Gene Ontology Consortium, www.geneontology.org.