A Framework for Ontology-Driven Subspace Clustering

Jinze Liu, Wei Wang
Department of Computer Science,
University of North Carolina at Chapel Hill
Chapel Hill,NC 27599, USA
{liuj, weiwang}@cs.unc.edu

Jiong Yang
Department of Computer Science,
University of Illinois
Urbana-Champaign, IL 61801, USA
jioyang@cs.uiuc.edu

ABSTRACT

Traditional clustering is a descriptive task that seeks to identify homogeneous groups of objects based on the values of their attributes. While domain knowledge is always the best way to justify clustering, few clustering algorithms have ever take domain knowledge into consideration. In this paper, the domain knowledge is represented by hierarchical ontology. We develop a framework by directly incorporating domain knowledge into clustering process, yielding a set of clusters with strong ontology implication. During the clustering process, ontology information is utilized to efficiently prune the exponential search space of the subspace clustering algorithms. Meanwhile, the algorithm generates automatical interpretation of the clustering result by mapping the natural hierarchical organized subspace clusters with significant categorical enrichment onto the ontology hierarchy. Our experiments on a set of gene expression data using gene ontology demonstrate that our pruning technique driven by ontology significantly improve the clustering performance with minimal degradation of the cluster quality. Meanwhile, many hierarchical organizations of gene clusters corresponding to a sub-hierarchies in gene ontology were also successfully captured.

Categories and Subject Descriptors: H.2.8 [Database Applications]: Data Mining H.2.8 [Database Applications]: Data Mining.

General Terms: Algorithms, Performance, Design.

Keywords: Subspace clustering, Ontology, Tendency Preserving.

1. INTRODUCTION

Clustering techniques have been studied extensively in statistics, pattern recognition, and machine learning. Clustering in high dimensional space is often problematic as theoretical results [5] questioned the meaning of closest matching in high dimensional spaces, which is known as the curse of dimensionality. Recent research work [16, 1, 2, 3, 6, 11] has focused on *subspace clustering*, discovering clusters embedded in the subspaces of a high dimensional data set.

Subspace clustering can be classified into two categories: partition-based algorithms and exhaustive enumeration algorithms, The PROCLUS [1] and the ORCLUS [2] algorithms partition the database into a given number of projected clusters based on representative clusters centering in subspaces. The information-theoretic co-clustering [8] approach simultaneously clusters the rows and columns to maximize their mutual information. A common feature of partition-based algorithm is that one object can appear in one and only one cluster. The other branch of subspace clustering algorithms exhaustively go through all subspaces potentially containing clusters and cluster in each subspace in a bottom-up fashion [3, 16, 12]. Compared with partition-based algorithms, one drawback of exhaustive subspace clustering is that its complexity is exponentially asymptotic to the dimensionality of the data space. In addition, it can generate a huge set of overlapping clusters due to the exponential number of unique subspaces. However, besides its capability of capturing all potential subspace clusters, the applicability of exhaustive approach to many areas can never be replaced by partition-based algorithm. The models of partition-based algorithms usually assume that one object only belongs to one cluster, which do not cater to the needs in many real-life applications. For example, a gene often participates in more than one gene activity and is often annotated with multiple function categories. Hence, the question with exhaustive subspace clustering algorithms is how to improve the scalability of the exhaustive subspace clustering algorithms while reducing the clusters into a small but relevant set.

Clustering analysis is purely syntactical in the sense that it does not take advantage of the existing knowledge in the learning process. Eventually, the most challenging problem is how to approach the matters of *interpretability*, i.e. why the objects in a cluster should be clustered together. In many applications, people may have significant amount of knowledge on the data set, which are usually utilized to measure the significance of a cluster. Traditionally, this knowledge is only used during the postprocessing step for validation of the clustering results. The following are some examples.

- Gene Expression Profiles. The gene expression profile is represented as a matrix where each row is a gene and each column is a condition while the corresponding entry records the expression level of the given gene under the given condition. A large number of gene expression profile analysis tools have been developed [16, 12]. However, all these work ignore one fact that there exists extensive amount knowledge of the genes. For instance, gene ontology (GO) [10] has been developed to categorize the relationship among genes. The GO can be used to identify biologically meaningful clusters.
- Customer Preference Profiles. In a user preference data set, each user (customer) may rank a set of goods. In reality, various goods are not independent of each other. For instance, VCR, DVD players, and VCD players are very similar while they are quite different from clothing and sports equipments. This type of knowledge could be utilized for analyzing the customer preferences.

KDD'04, August 22–25, 2004, Seattle, Washington, USA.

In this paper, we assume that the domain knowledge is captured in the ontology. The ontology is flexible yet powerful to capture the various degrees of relationship among objects (or attributes). In addition, it is used in many real applications. For example, in the bioinformatics community, the GO Consortium was formed to converge the efforts to make the controlled vocabulary of various genomic databases about diverse species in such a way that it can show the essential features shared by all the organisms [10].

We propose a hierarchical framework to directly incorporate the ontology knowledge into subspace clustering process. Our particular interest lies in searching subspace clusters that can be well explained by its ontology categories. However, is there a natural correspondence between the hierarchy of subspace clusters and the hierarchy of ontology? To answer this question, we give the following example.

EXAMPLE 1.1. *Table 1 presents a subset of zoo data in UCI KDD repository.*

animals	head	breaths	milk	legs	size	meat
squirrel	1	1	1	4	0	1
puma	1	1	1	4	1	0
dove	1	1	0	2	0	1
flamingo	1	1	0	2	1	1
perch	1	0	0	0	0	1
shark	1	0	1	0	1	0

Table 1: A database for a subset of zoo animals

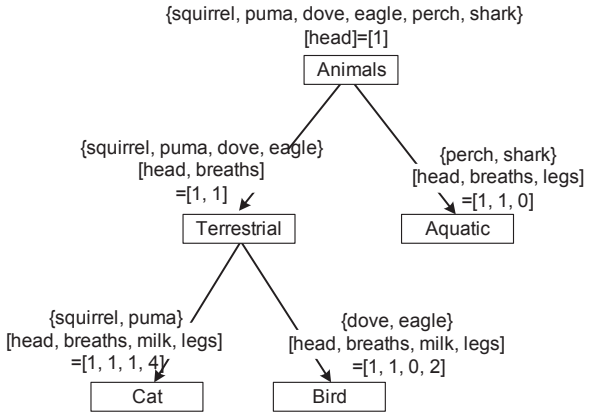


Figure 1: An animal ontology and subspace clusters corresponding to each category

A possible ontology for this small database is shown in Figure 1. Based on the ontology and the number of attributes shared by the animals at each ontology level, we observe that the higher level the category is in the hierarchy, the less attributes the objects in that category may share. For example, in each of "cat" and "bird" categories, the set of attributes$\{head, breaths, legs, milk\}$ have the same values among all the animals belonging to its category respectively, while all the animals in "terrestrial" category which includes both "cat" and "bird" share less attributes, i.e, $\{head, breaths\}$. The ontology can not only be used to guide the clustering process, but also can be used to validate the clustering results. If a cluster contains terms very far apart on the ontology hierarchy, then the cluster may not be very meaningful in that domain. Based on the above example, it is intuitive to see that Given an ontology hierarchy, the objects in the higher level of the category might share less attribute sets than the objects in the lower level of the hierarchy, which is a natural correspondence of the arrangement of subspace clusters along the subspace hierarchy.

Based on the above observation, given a database with a set of objects featuring a set of attributes, it will be interesting to find out which subset of objects can be clustered together over which subset attributes that can be classified into the same category located in the ontology hierarchy. We also want to find out for each category, which subset of attributes might contribute to the split of the object sets into more detailed classification categories.

We create a general framework for ontology-driven subspace clustering. This framework can be most beneficial for the hierarchically organized subspace clustering algorithm and ontology hierarchy, i.e., it is independent of the clustering algorithms and ontology application domain. To demonstrate the usefulness of this framework, we choose TP-cluster algorithm [12] and the gene ontology as two representatives of exhaustive subspace clustering and ontology respectively. Both of them have been proven useful in clustering gene expression profiles and gene function annotation.

Contribution

- We formally define an ontology hierarchy. Based on this, we use a substructure of the ontology hierarchy to interpret the categorical meaning of a cluster.
- We build a framework to incorporate domain knowledge (represented as ontology) into subspace clustering. This novel clustering algorithm automatically generates meaningful clusters (with respect to the ontology) while improving the performance.
- We design a new model to assess the objects' distribution of each ontology category in a cluster. Based on this, we developed a ontology-based pruning technique to minimize the redundancy in the subspace clusters.
- Our experiment results demonstrate that the ontology paths are well corresponded to certain local structure of hierarchically organized subspace clusters. Meanwhile, the performance of ontology-driven subspace clustering algorithm has great improvement with minimum loss of clustering quality.

The remainder of the paper is organized as follows. Section 2 defines the model of Ontology and Tendency Preserving cluster. Section 3 presents the ontology-driven subspace clustering algorithm in detail. An extensive performance study on Microarray data is reported in Section 4. Section 5 concludes the paper and discusses some future work.

2. MODEL

2.1 Ontology Framework

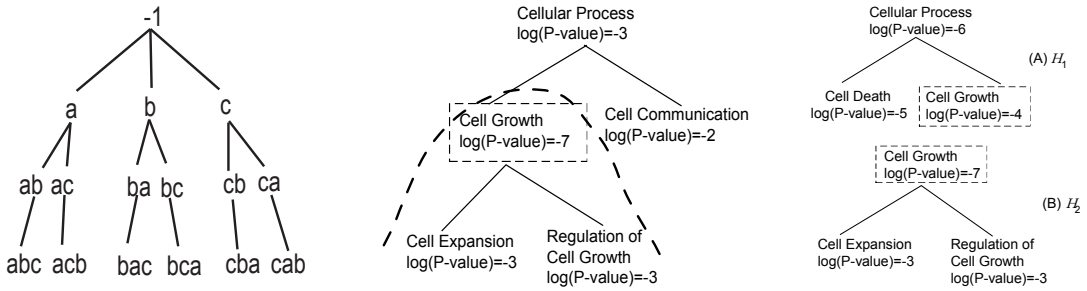
We start with the formal definition of the ontology.

DEFINITION 2.1. *An ontology is a sign system $\mathcal{O}$:=$(\mathcal{L}, \mathcal{H}, \mathcal{R})$, which consists of*

- *A lexicon: The lexicon $\mathcal{L}$ contains a set of natural language terms.*
- *A hierarchy $\mathcal{H}$: Terms in $\mathcal{L}$ are taxonomically related by the directed , acyclic, transitive, reflexive relation $\mathcal{H}$. ($\mathcal{H} \subset \mathcal{L} \times \mathcal{L}$);*

A top term $\mathcal{R} \in \mathcal{L}$. For all $l \in \mathcal{L}$, it holds: $\mathcal{H}(l, \mathcal{R})$.

The ontology essentially defines a hierarchy where each node corresponds to a *lexicon* or a categorical term. Each categorical term contains a set of objects and the set of objects in a descendent term is always a subset of the objects in its ancestor category. Figure 1 presents an example of animal ontology.



(a) A complete TP-Cluster tree $\mathcal{A}$=$\{a, b, c\}$

(b) An example of OST (under the curve)

(c) Two OSTs $\mathcal{H}_2 \preceq \mathcal{H}_1$.

Figure 2: 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.

2.2 Tendency Preserving Cluster Model

Let $\mathcal{D}$ be a database containing the object set $\mathcal{O}$ under a set of attributes $\mathcal{A}$. The whole database can be represented in a data matrix $\mathcal{M}$, where M_{ij} is the entry value of object i under attributes j ($0 < i \leq |\mathcal{O}|, 0 < j \leq |\mathcal{A}|$).

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

DEFINITION 2.2. *Let $\mathcal{O}$ be a subset of objects in the database $\mathcal{D}$, $\mathcal{O} \subseteq \mathcal{D}$. Let $\mathcal{T}$ be a subset of attributes, $\mathcal{T} \subseteq \mathcal{A}$. Let $\mathcal{R}$: $\mathcal{T} \times \mathcal{O} \times 2^{|\mathcal{A}|} \rightarrow \mathcal{I}$ be the function that assigns the rank of an object i's attribute j to be r, if the expression value of the object i under attributes j is the r^{th} 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 2.2 first defines the rank function $\mathcal{R}$. Based on the rank function, a TP-Cluster is defined as a maximum subset of objects which have consistent ranks along a subset of attributes.

2.3 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 (a), where each TP-Cluster is represented by a sequence.

DEFINITION 2.3. *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.

2.4 Annotation of a Cluster by Ontology

The hypergeometric distribution is used to model the probability of observing at least k objects from a cluster of n objects by chance in a category containing f objects from a total database size of g objects. 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 objects from a particular category to a greater extent than that which would be expected by chance. If the majority of objects in a cluster belong to the same category, then it is unlikely that this happens by chance and the category's P-value would be close to 0.

We use an appropriate subtree in the ontology hierarchy to annotate a cluster. The subtree is rooted at the node of the most significant category and includes all of its significant reachable subcategories.

DEFINITION 2.4. *Given a cluster $\mathcal{C}$, its significant 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 $e' \in \mathcal{E}'$.*

Figure 2 (b) shows a set of significant GO function categories of a gene cluster organized in a tree structure. To determine the OST representing this cluster, we first find 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 2.5. *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 2 (c) contains two gene 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 $\mathcal{H}_1$.

Problem Statement: Let $\mathcal{D}$ be a database with object set $\mathcal{O}$ and attribute set $\mathcal{A}$. Given a threshold θ_p for category enrichment and the ontology hierarchy, our goal is to extract a hierarchy of enriched TP-Clusters consistent with the whole or partial ontology hierarchy.

3. CONSTRUCTION OF ONTOLOGY RELEVANT TP-CLUSTER TREE

In this section, we present the algorithm to build an ontology relevant TP-Cluster tree.

3.1 Construction of the TP-Cluster tree

The TP-clustering process can be summarized in two steps:

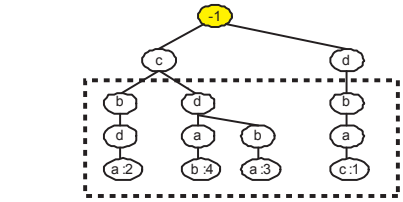
1. **Preprocess the data**. Each row in the data matrix will be converted to an ordered sequence of column labels corresponding to increasing entry values. Those sequences will be the inputs to the next step. An initial prefix tree containing the sequence of every object in the database will be constructed.The ontology information of genes is feeded into the initial tree at the root level.

2. **Depth-first traversal to develop ontology relevant TP-Cluster**. The initial prefix tree is recursively visited in the depth-first order. During each node visit, the cluster corresponding to that node is evaluated against ontology enrichment and ontology consistency. Once it passes the ontology assessment, further development of its subtree are processed by suffix concatenation. Otherwise, its subtree will be pruned.

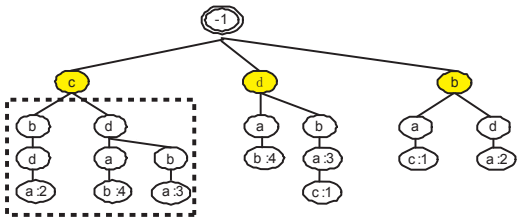
gID	*a*	*b*	*c*	*d*	*sequence*
1	4002	284	4108	228	*dbac*
2	401	281	120	298	*cbda*
3	401	292	109	238	*cdba*
4	280	318	37	215	*cdab*

Table 2: An example dataset.

We use the dataset in Table 2 in the following example to illustrate the suffix concatenation step during the tree construction process.



(A) Initial tree



(B) First suffix concatenations at level 1

Figure 3: The illustration of suffix tree concatenation.

EXAMPLE 3.1. *For sequences in Table 2, the initial prefix tree representing the whole database is presented in Figure 3 (A) and the suffix concatenation upon visiting the first node "-1" is illustrated in Figure 3 (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 3 (B), the suffixes to be inserted to "-1"'s subtree are those inside the rectangle box shown in Figure 3 (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 3(B) are the next set of suffixes to be inserted.

3.2 Ontology-based Pruning Techniques

The ontology information serves the following two purposes: (1) the assessment of category enrichments of a cluster. (2) the guidance to select the subset of attributes critical to a category. These two functionalities of ontology information are transformed into two pruning techniques in the ONTP-clustering algorithm.

The first pruning technique is based on the distribution of objects in different categories in a cluster. Since the first one focuses on the computation of P-value in each category, we omit the detailed description and only explain the second technique, which is to use OST extracted in a parent cluster to guide the selection of its descendent TP-Cluster clusters, by favoring ontology relevant children clusters defined in Definition 3.1. Our criterion is based on the hypothesis that, the TP-Clusters in the higher dimensional space are enriched in more specific categories.

DEFINITION 3.1. *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 an ontology relevant descendent of $\mathcal{C}$ if $\mathcal{H}' \preceq \mathcal{H}$.*

CRITERION 3.1. *Let $\mathcal{C}$ be a TP-Cluster and $\mathcal{C}'$ be one of $\mathcal{C}$'s descendants, we say the development of $\mathcal{C}'$ is not viable if it is not an ontology relevant descendent of $\mathcal{C}$.*

We present the ONTP-clustering algorithm of extracting ontology relevant TP-Clusters in Algorithm *smartGrowTree*.

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 ONTP-clustering construction: For ONTP-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/object 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 ONTP-clustering. The techniques based on ontology knowledge further prune the potential clusters. Both the time and space complexities of the two algorithms are exponential determined by the nature of being an NP-hard problem. The pruning effects are largely determined by the relationship between a TP-Cluster and the significance of its underlying functional categories.

4. EVALUATION

Our experiments demonstrate the applicability of ONTP-clustering algorithm to clustering biologically related genes with effective pruning techniques based on GO. We denote the algorithm without ontology-based pruning as TP-clustering and the one with prunning as TP-Cluster tree. The results are evaluated through the comparison of TP-clustering and TP-Cluster tree 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.

Our 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 [10] 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 3: Statistics for the three categories.

The first set of experiments was done using the ONTP-clustering algorithm and cellular component ontology category to evaluate the performance by varying parameters n_r and θ_p. As shown in Figure 4 a), the response time of the ONTP-clustering algorithm decreases as the significance threshold decreases and as the minimum number of rows increases. 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 4 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 3.1. The percentage of not enriched cluster increases significantly as θ_p decreases. It also explains the performance gain of ONTP-clustering at the same time. Also the percentage of clusters being pruned due to Criterion 3.1 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 experiments in Figure 4 c) is a comparison between ONTP-clustering algorithm and TP-clustering algorithm. For each algorithm, we have done two tests with different settings of n_r. ONTP-clustering algorithm consistently outperforms TP-clustering especially when θ_p is relatively low. The response time of ONTP-clustering can be as short as 1/4 of that of the TP-clustering algorithm. The TP-clustering algorithm generates a large number of TP-Clusters, of which only 10% are enriched when $\theta_p = -5$. Compared with TP-clustering, ONTP-clustering generates less than half of the number of TP-Clusters and almost the same number of enriched TP-Clusters. Overall, ONTP-clustering improves the performance with minimum loss of the enriched clusters.

Figure 4 d) 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 3. 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 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.

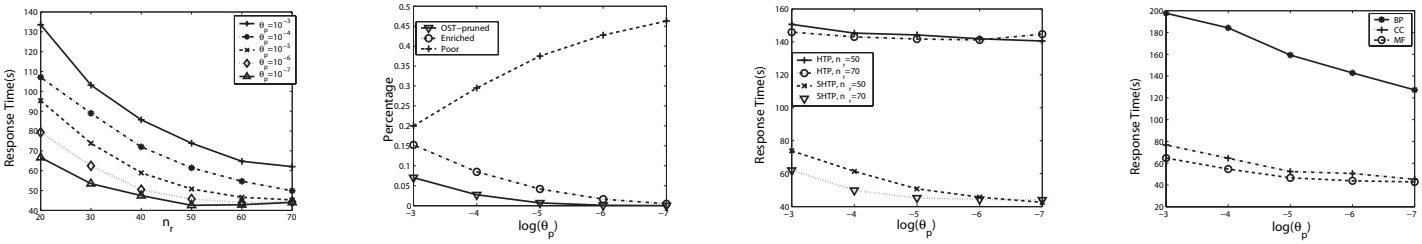
4.1 Mapping between GO and the TP-Cluster Tree

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

In Figure 5, (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 → DNA metabolism → DNA repair".

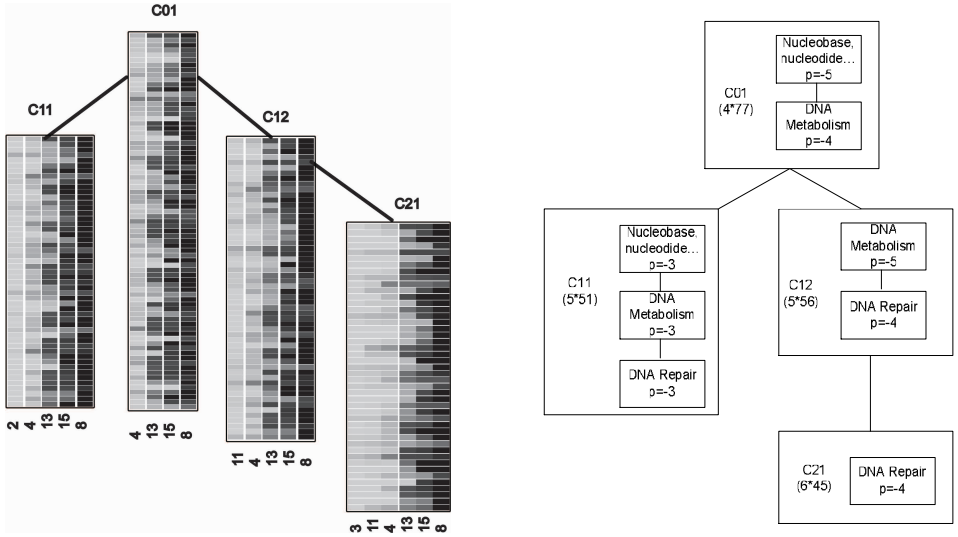
The root cluster C_{01} in (A) is the largest cluster with 71 genes. However, it has the smallest number of conditions shared by all genes in its cluster, i.e. $(4, 15, 13, 8)$. Its OST shown at the top of the hierarchy in (B) is rooted at the category, Necleoside. As we go down the hierarchy of clusters 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 the a even deeper function group, i.e. "DNA repair".

This example illustrates the connection between the ontology hierarchy and the TP-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.



a) Response Time b)Clusters Distribution c) Response time d)

Figure 4: The performance of the ONTP-clustering varying n_r and θ_p.



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

Figure 5: 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.

5. CONCLUSIONS AND FUTURE WORK

The clustering analysis is traditionally syntactical in the sense that it does not take advantage of the existing knowledge in the learning process. In the paper, we present a general framework by incorporating ontology hierarchy into the process of hierarchical subspace clustering. We extract the set of ontology explainable clusters during the clustering process in one run. Meanwhile, we also discuss an ontology pruning technique that enhances clustering algorithm and clustering result. Our experiments on yeast gene expression data demonstrates the effectiveness of ontology-based pruning. We also successfully extract the partial ontology hierarchy from the subspace clusters. Our future work will use clustering results and domain ontology for efficient and effective classification for the unknown objects.

6. REFERENCES

[1] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD*, 1999.

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *SIGMOD*, pages 70-81, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD*, 1998.

[4] 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.

[5] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbors meaningful. In *Proc. of the Int. Conf. Database Theories*, pages 217-235, 1999.

[6] C. H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *SIGKDD*, pages 84-93, 1999.

[7] Y. Cheng and G. Church. Biclustering of expression data. In *Proc. of 8th International Conference on Intelligent System for Molecular Biology*, 2000.

[8] I. S. Dhillon, Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning. In *SIGKDD*, 2001.

[9] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-bsed algorithm for discovering clusters in large spatial databases with noise. In *SIGKDD*, pages 226-231, 1996.

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

[11] H.V.Jagadish, J.Madar, and R. Ng. Semantic compression and pattern extraction with fasicicles. In *VLDB*, pages 186-196, 1999.

[12] J.Liu and W.Wang. Flexible clustering by tendency in high dimensional spaces. Technical Report TR03-009, Computer Science Department, UNC-CH, 2003.

[13] 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 saccharomyces cerevisiae by microaray hybidization. *Molecular Biology of the Cell*, 9:3273-3297, 1998.

[14] S.Tavazoie, Jason D.Hughes, M.J.Campbell, R.J.Cho, G.M.Church. Systematic determination of genetic network architecture, *Nature Genetics*, 22: 281-285, 1999.

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

[16] H. Wang, W. Wang, J. Yang, and P. Yu. Clustering by pattern similarity in large data sets, in *SIGMOD*, pp. 394-405, 2002.