Subspace Clustering of Microarray Data based on Domain Transformation

Jongeun Jun[1], Seokkyung Chung[2]*, and Dennis McLeod[1]

[1] Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA
[2] Yahoo! Inc., 2821 Mission College Blvd, Santa Clara, CA 95054, USA
jongeunj@usc.edu, *schung@yahoo-inc.com*, *mcleod@usc.edu*

Abstract. We propose a mining framework that supports the identification of useful knowledge based on data clustering. With the recent advancement of microarray technologies, we focus our attention on gene expression datasets mining. In particular, given that genes are often co-expressed under subsets of experimental conditions, we present a novel subspace clustering algorithm. In contrast to previous approaches, our method is based on the observation that the number of subspace clusters is related with the number of maximal subspace clusters to which any gene pair can belong. By performing discretization to gene expression profiles, the similarity between two genes is transformed as a sequence of symbols that represents the maximal subspace cluster for the gene pair. This domain transformation (from genes into gene-gene relations) allows us to make the number of possible subspace clusters dependent on the number of genes. Based on the symbolic representations of genes, we present an efficient subspace clustering algorithm that is scalable to the number of dimensions. In addition, the running time can be drastically reduced by utilizing inverted index and pruning non-interesting subspaces. Experimental results indicate that the proposed method efficiently identifies co-expressed gene subspace clusters for a yeast cell cycle dataset.

1 Introduction

With the recent advancement of DNA microarray technologies, the expression levels of thousands of genes can be measured simultaneously. The obtained data are usually organized as a matrix (also known as a gene expression profile), which consists of m columns and n rows. The rows represent genes (usually genes of the whole genome), and the columns correspond to the samples (e.g. various tissues, experimental conditions, or time points).

Given this rich amount of gene expression data, the goal of microarray analysis is to extract hidden knowledge (e.g., similarity or dependency between genes) from this matrix. The analysis of gene expressions may identify mechanisms of gene regulation and interaction, which can be used to understand a function of a

* To whom correspondence should be addressed. Part of this research was conducted when the author was at University of Southern California.

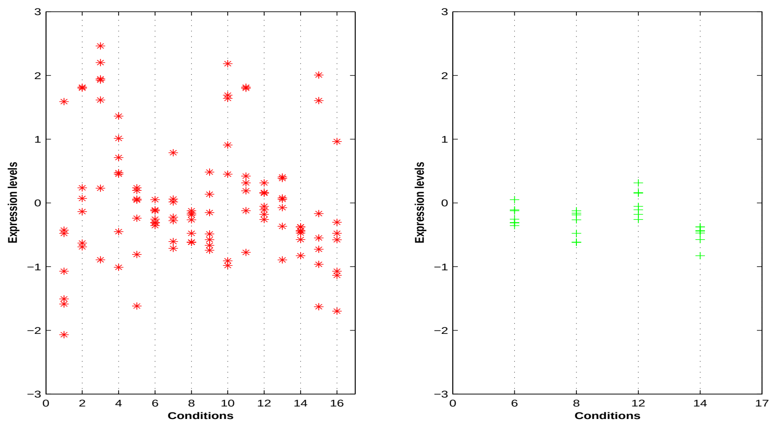


Fig. 1. Plot of sample gene expression data across whole conditions (shown in left hand side) versus subset of conditions (shown in right hand side)

cell [6]. Moreover, comparison between expressions in a diseased tissue and a normal tissue will further enhance our understanding in the disease pathology [7]. Therefore, data mining, which transforms a raw dataset into useful higher-level knowledge, becomes a must in bioinformatics.

One of the key steps in gene expression analysis is to perform clustering genes that show similar patterns. By identifying a set of gene clusters, we can hypothesize that the genes clustered together tend to be functionally related. Traditional clustering algorithms have been designed to identify clusters in the full dimensional space rather than subsets of dimensions [6, 14, 4, 11]. When correlations among genes are not apparently visible across the whole dimensions as shown in the left-side graph of Figure 1, the traditional approaches fail to detect clusters. However, it is well-known that genes can manifest a coherent pattern under subsets of experimental conditions as shown in the right-side graph of Figure 1. Therefore, it is essential to identify such local patterns in microarray datasets, which is a key to revealing biologically meaningful clusters.

In this paper, we propose a mining framework that supports the identification of meaningful subspace clusters. When m (the number of dimensions) is equal to 50 (m for gene expression data usually varies from 20 to 100), the number of possible subspaces is $2^{50} - 1 \approx 1.1259 \times 10^{15}$. Thus, it is computationally expensive to search all subspaces to identify clusters. To cope with this problem, many subspace clustering algorithms first identify clusters in low dimensional spaces, and use them to find clusters in higher dimensional spaces based on *apriori* principle [1] [3]. However, this approach is not scalable to the number of dimensions in general.

[3] If a collection of points C form a dense cluster in a k-dimensional space, then any subset of C should form a dense cluster in a $(k-1)$-dimensional space.

Notation	Meaning
n	The total number of genes in gene expression data
m	The total number of dimensions in gene expression data
X	$n \times m$ gene expression profile matrix
x_i	The i-th gene
x_{ij}	The value of x_i in j-th dimension
S_i	The set of symbols for i-th dimension ($S_i \cap S_j = \emptyset$ if $i \neq j$)
s_{ij}	A symbol for x_{ij}
s_i	A sequence of symbols for x_i
K	The maximum number of symbols
p_i	A pattern
mp_{ij}	The maximal pattern for x_i and x_j
P_a	A set of maximal patterns that contains a symbol a
P_i	The set of all maximal patterns in i-dimensional subspace
P	The set of all maximal patterns in X
α_i	The minimum number of elements that an i-dimensional subspace cluster should have
β	The minimum dimension of subspace clusters

Table 1. Notations for subspace clustering

In contrast to the previous approaches, our method is based on the observation that the maximum number of subspaces is related with the number of maximal subspace clusters to which any two genes can belong. In particular, by performing discretization to gene expression profiles, we can transform the similarity between two genes as a sequence of symbols that represent the maximal subspace cluster for the gene pairs. This transformation allows us to limit the number of possible subspace clusters to $\frac{n(n-1)}{2}$ where n is the number of genes. Based on the transformed data, we present an efficient subspace clustering algorithm that is scalable to the number of dimensions. Moreover, by utilizing inverted index and pruning (even conservatively) non-interesting subspace clusters, the running time can be drastically reduced. Note that the presented clustering algorithm can detect subspace clusters regardless of whether their coordinate values are consecutive to each other or not.

The remainder of this paper is organized as follows. In Section 2, we explain the proposed subspace clustering algorithm. Section 3 presents experimental results. In Section 4, we briefly review the related work. Finally, we conclude the paper and provide our future plans in Section 5.

2 The Subspace Clustering Algorithm

In this section, we present the details of each step of our approach. Table 1 shows the notations, which will be used throughout this paper.

The first step of our clustering is to quantize gene expression dataset (Section 2.1). The primary reason for discretization is that we need to efficiently extract a maximal subspace cluster to which every gene pair belongs. In-depth discussions on why we need discretization, and details of our algorithm are presented in Section 2.2. In Section 2.3, we explain how we can further reduce computational cost by utilizing inverted index and pruning. Finally, in Section 2.4, we explain how to select meaningful subspace clusters. Figure 2 sketches the proposed subspace clustering algorithm.

2.1 Discretization

In general, discretization approach can be categorized into three ways:

- *Equi-width bins.* Each bin has approximately same size.
- *Equi-depth bins.* Each bin has approximately same number of data elements.
- *Homogeneity-based bins.* The data elements in each bin are similar to each other.

In this paper, we use a homogeneity-based bins approach. In particular, we utilize K-means clustering with Euclidean distance metric [5]. Because we apply K-means to 1-dimensional data, each cluster corresponds to the interval. Additionally, K corresponds to the number of symbols for discretization.

Once we identify clusters, genes belonging to a same cluster are discretized with a same symbol. That is, x_{id} and x_{jd} are represented as a same symbol if and only if $x_{id}, x_{jd} \in C_{kd}$ where C_{kd} is k-th cluster in d-th dimension. The complexity for this step is $O(nmK)$ where n and m correspond to the number of genes and conditions, respectively. In this paper, we use same value of K across all dimensions for simplicity.[4] However, in Section 3, we investigate the effect of different value of K in terms of running time.

2.2 Identification of Candidate Subspace Clusters based on Domain Transformation

The core of our approach lies in domain transformation to tackle subspace clustering. That is, the problem of subspace clustering is significantly simplified by transforming dataset into domain of gene-gene relations (Figure 3). In this Section, we explore how to identify candidate subspace clusters based on the notion of domain transformation. Before we present detailed discussions on the proposed clustering algorithm, definitions for basic terminology are provided first.

Definition 1 Induced-string. A string $string_1$ is referred to as an induced-string of $string_2$ if every symbol of $string_1$ appears in $string_2$ and length of $string_1$ is less than $string_2$.

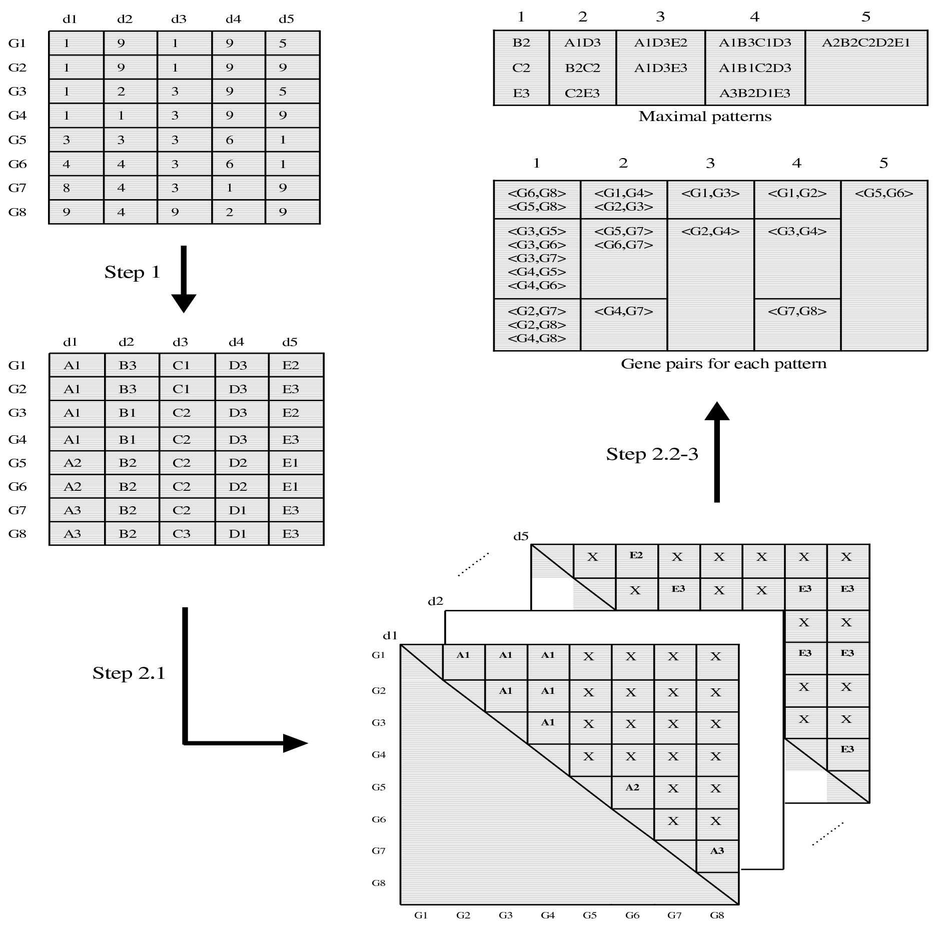
[4] More effective discretization algorithm for gene expression data is currently under development.

```
Step 1. Discretization and symbolization of gene expression data:
  Initialization: $i$= 1.
  1.1 Perform $K$-means clustering on $i$-th dimension.
  1.2 Using $S_i$, assign symbols to each cluster.
  1.3 $i = i + 1$.
  Repeat Step 1 for $i \le m$.

Step 2. Enumeration of all possible maximal patterns for each gene pair
        and construction of inverted index for each dimension and Hash Table:
  2.1 Extract a maximal pattern ($mp_{ij}$) between $x_i$ and $x_j$ using $s_i$ and $s_j$.
  2.2 Insert $mp_{ij}$ to inverted index at dimension $k$ if the length of
      $mp_{ij}$ is equal to $k$.
  2.3 Insert the gene pair $x_i$ and $x_j$ to Hash Table indexed by $mp_{ij}$.
  Repeat Step 2 for all gene pairs.

Step 3. Computation of support, pruning and generating candidate clusters:
  Initialization: $i = \beta$ where $\beta$ is a user-defined threshold for the
                  minimum dimension of a subspace cluster.
  3.1 Compute support of each maximal pattern ($p$) at $i$ using inverted index.
  3.2 Prune $p$ at $i$ and super-pattern of $p$ at $i + 1$ if support of $p$ is less than $\alpha_i$.
  3.3 $i = i + 1$
  Repeat Step 3 for $i \le m - 1$.

Step 4. Identification of interesting subspace clusters:
  Initialization: $i = \beta$.
  4.1 For each pattern ($p$) of candidate clusters at $i$, calculate difference between
      its support and support of each super-pattern of $p$ at $i + 1$, and assign
      the biggest difference as maximum difference of $p$.
  4.2 Remove $p$ from candidate clusters at $i$ if maximum difference of $p$ is less
      than user-defined threshold.
  4.3 $i = i + 1$
  Repeat Step 4 for $i \le m - 1$.
```

Fig. 2. An overview of the proposed clustering algorithm

Definition 2 (Maximal) pattern. Given s_i and s_j, which are symbolic representations of x_i and x_j, respectively, any induced-string of both s_i and s_j is referred to as a pattern of x_i and x_j. A pattern is referred to as a maximal pattern of x_i and x_j (denoted as mp_{ij}) if and only if every pattern of x_i and x_j (except mp_{ij}) is an induced-string of mp_{ij}.

For example, if $s_1 = a_1b_1c_2d_3$ and $s_2 = a_1b_1c_1d_3$, then maximal pattern for x_1 and x_2 is $a_1b_1d_3$. Since the length of the maximal pattern is 3, the maximum



	d1	d2	d3	d4	d5
G1	1	9	1	9	5
G2	1	9	1	9	9
G3	1	2	3	9	5
G4	1	1	3	9	9
G5	3	3	3	6	1
G6	4	4	3	6	1
G7	8	4	3	1	9
G8	9	4	9	2	9

	d1	d2	d3	d4	d5
G1	A1	B3	C1	D3	E2
G2	A1	B3	C1	D3	E3
G3	A1	B1	C2	D3	E2
G4	A1	B1	C2	D3	E3
G5	A2	B2	C2	D2	E1
G6	A2	B2	C2	D2	E1
G7	A3	B2	C2	D1	E3
G8	A3	B2	C3	D1	E3

1	2	3	4	5
B2	A1D3	A1D3E2	A1B3C1D3	A2B2C2D2E1
C2	B2C2	A1D3E3	A1B1C2D3	
E3	C2E3		A3B2D1E3	

Maximal patterns

1	2	3	4	5
<G6,G8> <G5,G8>	<G1,G4> <G2,G3>	<G1,G3>	<G1,G2>	<G5,G6>
<G3,G5> <G3,G6> <G3,G7> <G4,G5> <G4,G6>	<G5,G7> <G6,G7>	<G2,G4>	<G3,G4>	
<G2,G7> <G2,G8> <G4,G8>	<G4,G7>		<G7,G8>	

Gene pairs for each pattern

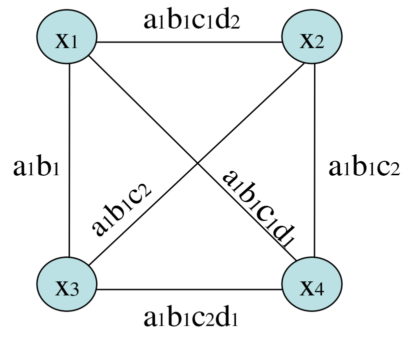
Fig. 3. An illustration of domain transformation

dimensions of candidate subspace cluster that can host x_1 and x_2 is 3. Thus, the set of genes that have same maximal pattern can be a candidate maximal subspace cluster. By discretizing gene expression data, we can obtain the upper bound for the number of maximal patterns, which is equal to $\frac{n(n-1)}{2}$.

Because a pattern is represented as a form of string, based on the definition of induced-string, the notion of super-pattern is defined as follows:

Definition 3 Super-pattern. A pattern p_i is a super-pattern of p_j if and only if p_j is an induced string of p_i.

Figure 4 illustrates how we approach the problem of subspace clustering. Each vertex represents a gene, and the edge between vertexes shows a maximal pattern between two genes. In order to find meaningful subspace clusters, we define the minimum number of objects (α_i) that an i-dimensional subspace cluster



	2	3	4
a_1b_1	$<x_1\ x_3>$	$<x_2\ x_3>$ $<x_2\ x_4>$	$<x_3\ x_4>$ $<x_1\ x_4>$ $<x_1\ x_2>$
$a_1b_1c_2$		$<x_2\ x_3>$ $<x_2\ x_4>$	$<x_3\ x_4>$
$a_1b_1c_1d_2$			$<x_1\ x_2>$
$a_1b_1c_1d_1$			$<x_1\ x_4>$
$a_1b_1c_2d_1$			$<x_3\ x_4>$

Fig. 4. A sample example of subspace cluster discovery

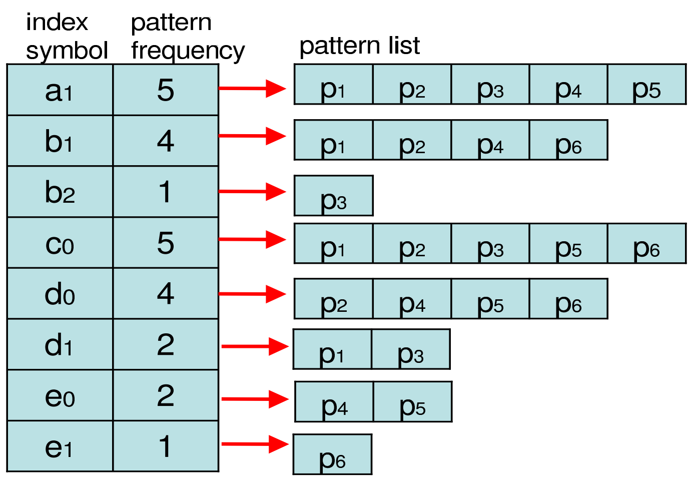
should have. For this purpose, the notion of support is defined as follows:

Definition 4 Support. Given a pattern p_k, support of p_k is the total number of gene pairs (x_i and x_j) such that p_k is an induced-string of mp_{ij} or p_k is a maximal pattern of x_i and x_j.

Thus, the problem of subspace clustering is reduced to computing support of each maximal pattern, and identifying maximal patterns whose support exceeds α_i if the length of maximal pattern is equal to i. The maximal pattern between x_1 and x_3 is defined as a_1b_1 in Figure 4. This means that x_1 and x_3 has potential to be grouped together in 2-dimensional subspace. However, we need to consider whether there exists enough number of gene pairs that has a pattern a_1b_1. This step can be achieved by computing support for a_1b_1, and checking whether the support exceeds α_2 or not. In this example, support for a_1b_1 is 6.

2.3 Efficient Super-pattern Search based on Inverted Index

Achieving an efficient super-pattern search for support computation is important in our algorithm. The simple approach to searching super-patterns for a given maximal pattern at dimension d is to conduct a sequential scan on all maximal patterns at dimension d' ($d' > d$). The following shows the running time for this approach.



$p_1 = a_1b_1c_0d_1$; $p_2 = a_1b_1c_0d_0$; $p_3 = a_1b_2c_0d_1$
$p_4 = a_1b_1d_0e_0$; $p_5 = a_1c_0d_0e_0$; $p_6 = b_1c_0d_0e_1$

Fig. 5. Inverted index at 4-dimensional subspace

$$\sum_{i=\beta}^{n-1} |P_i| \sum_{j=i+1}^{m} (j \times |P_j|) \tag{1}$$

where $|P_j|$ is the number of maximal patterns at dimension j and β is the minimum dimension of subspace clusters.

However, we can reduce the time by utilizing inverted index, which has been widely used in modern information retrieval. In inverted index [10], the index associates a set of documents with terms. That is, for each term t_i, we build a document list (D_i) that contains all documents containing t_i. Thus, when a query q is composed of $t_1, ..., t_k$, to identify a set of documents which contains q, it is sufficient to examine the documents that contain all t_i's (i.e., intersection of D_i's) instead of checking whole document dataset. By implementing each document list as a hash table, looking up documents that contain t_i takes constant time.

In our super-pattern search problem, each term and document correspond to symbol and pattern, respectively. Figure 5 illustrates the idea. Each pattern list is also implemented as a hash table. When a query pattern is composed of multiple symbols, symbols are sorted (in ascending order) according to their pattern frequency. After then, we start to take intersection of pattern lists whose pattern frequency is lowest. For example, given a query pattern $b_2c_0d_0$, we search super pattern as follows: [5]

[5] If we reach an empty set while taking intersection, then there is no need to keep intersection of pattern lists.

$$(P_{b_2} \cap P_{d_0}) \cap P_{c_0} = (\{p_3\} \cap \{p_2, p_4, p_5, p_6\}) \cap P_{c_0} = \emptyset$$

where P_a is a pattern list for a symbol a.

By taking intersection of pattern list whose size is small, we can reduce the number of operations. The worst time complexity for the identification of all super-patterns for all query patterns ($a = (a_1, a_2..., a_k)$) in P_k is shown as follows:

$$T_k = \sum_{i=k+1}^{m} \sum_{a \in P_k} (min(|P^i_{a_1}|, ..., |P^i_{a_k}|) \times (k-1)) \tag{2}$$

where $|P^i_{a_k}|$ corresponds to the length of pattern list for the symbol a_k at dimension i.

Note that the worst time complexity for the construction of inverted index at each dimension k takes $k \times |P_k|$. Thus, the total time complexity for super-pattern search is given as below:

$$\sum_{k=\beta}^{m} (k \times |P_k|) + \sum_{k=\beta}^{m-1} (klogk + |P_k| \times T_k) \tag{3}$$

Moreover, we can further reduce running time through the pruning step. In particular, we use the notion of expected support in Zaki *et al.* [15]. That is, if support for p_{ij} in dimension k is less than α_k, then besides eliminating p_{ij} in future consideration of subspace clustering, we do not consider super-patterns of p_{ij} in dimension $k+1$ any more.

2.4 Identification of Meaningful Subspace Clusters

After pruning maximal patterns using the notion of support, the final step is to identify *meaningful* subspace clusters, which corresponds to the Step 4 in Figure 2. Since we are interested in maximal subspace clusters, we scan maximal patterns (denoted as p) at dimension i, and compare support of p with support of all super-patterns (denoted as p') at dimension $(i+1)$. Any maximal pattern at i (p) are not considered for a subspace cluster if the difference between support of p and that of p' is less than user-defined threshold.

We illustrate this step with Figure 6. As shown, there are three maximal patterns at dimension 2 (A1B1, A1C1 and A1C2) and two maximal patterns at dimension 3 (A1B1C1 and A1B1C2), respectively. The cluster with a pattern A1B1 (C_1) contains the clusters with a pattern A1B1C1 (C_2) and A1B1C2 (C_3). If the support between C_1 and C_2 (or the support between C_1 and C_3) is less than user-defined threshold, then C_1 is ignored and both C_2 and C_3 are kept as meaningful subspace clusters. Otherwise, C_1 is also considered as a meaningful cluster besides C_2 and C_3.

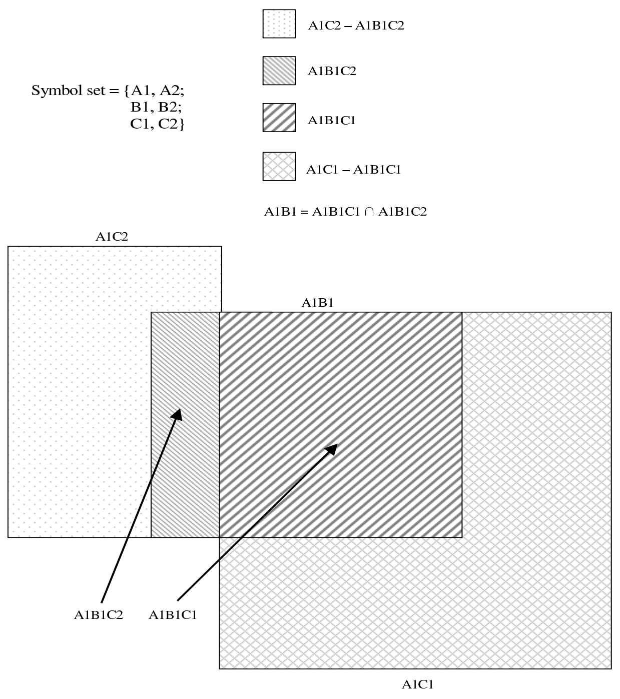


Fig. 6. Identification of subspace clusters

3 Experimental Results

In this section, we present experimental results that demonstrate the effectiveness of the proposed clustering algorithm. Section 3.1 illustrates our experimental setup. Experimental results are presented in Section 3.2.

3.1 Experimental Setup

For empirical evaluation, the proposed clustering algorithm was tested on yeast cell cycle data. The data is a subset from the original 6,220 genes with 17 time points listed by [3]. Cho *et al.* sampled 17 time points at 10 minutes time interval, covering nearly two full cell cycles of yeast *Saccharomyces cerevisiae*. Among 6,220 genes, 2,321 genes were selected based on the largest variance in their expression. In addition, one abnormal time point was removed from the data set as suggested by [11], consequently, the resulting data consists of 2,321 genes with 16 time points. Our analysis is primarily focused on this data set.

3.2 Experimental Results

Figure 7 demonstrates how well our clustering algorithm was able to capture highly correlated clusters under a subset of conditions. The x-axis represents the conditions, and the y-axis represents expression level. As shown, subspace clusters do not necessarily manifest high correlations across all conditions. We also verified biological meanings of subspace clusters with the Gene Ontology [2]. For example, YBR279w and YLR068w (Figure 7(b)) have the most specific common parent "nucleobase, nucleoside, nucleotide and nucleic acid metabolism" in the Gene Ontology. However, they showed low correlation in full dimensions. In addition, YBR083w and YDL197C (Figure 7(d)) belonging to subspace cluster 625 participate in a similar biological function, "transcriptional control".

In order to investigate the number of maximal patterns in various scenarios, pattern rate, which represents how many maximal patterns are generated in comparison with the number of possible maximal patterns, is defined as follows:

$$Pattern\ Rate = \#maximal\ patterns\ /\ \frac{n(n-1)}{2} \tag{4}$$

Figure 8 shows relationships between the number of genes and pattern rate. The x-axis represents the number of genes (n), and the y-axis represents the pattern rate. As shown, pattern rate decreases as n increases. This property is related with the nature of datasets. In general, datasets should have certain amount of correlations among object pairs. Otherwise, clustering cannot detect meaningful clusters from the dataset. Thus, the increasing rate for the number of actual maximal patterns is much less than the increasing rate for possible number of maximal patterns ($\frac{n(n-1)}{2}$).

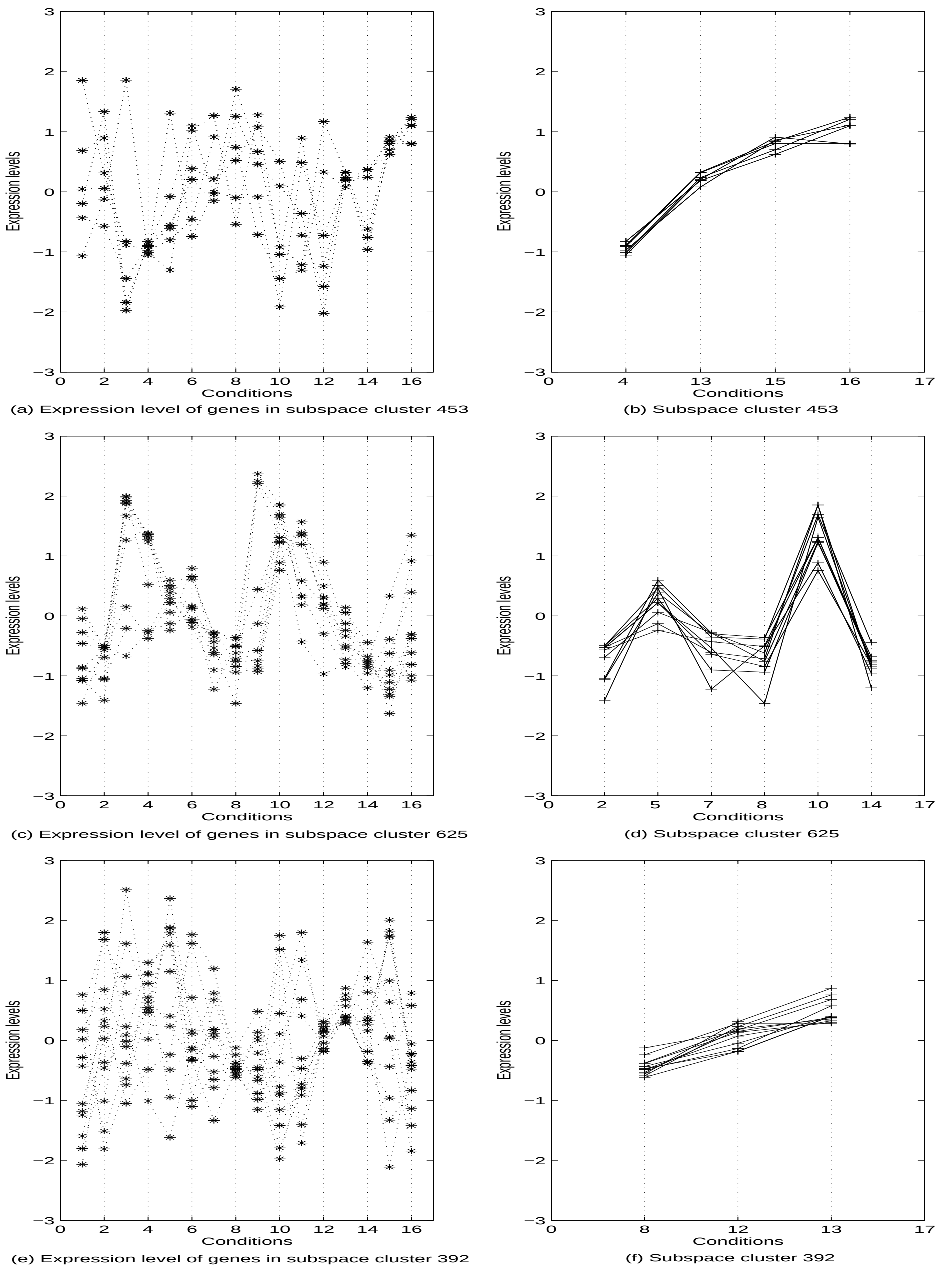
Figure 9 shows how much running time can be improved by using inverted index. The x-axis represents the number of symbols, and the y-axis represents the ratio of Equation (3) to Equation (1), which is defined as running time rate. As shown, we could observe significant improvement. For example, when the number of symbols is 7. the running time was reduced more than 20 times. Moreover, we could observe performance improvement as the number of symbols increases. This is primarily because the length of pattern list in inverted index decreases as the number of symbols increases.

Figure 10 shows the scalability of our algorithm in terms of the number of dimensions. The x-axis represents the number of dimensions, and the y-axis represents running time rate, which is defined by running time at d divided by running time when d is 4. As shown, we could observe linear increase in running time rate, which supports the scalability of our algorithm.

4 Related Work

In this section, we briefly discuss key differences between our work and previous approaches on subspace clustering. Jiang *et al.* [8] and Parsons *et al.* [9] provide the comprehensive review on gene expression clustering and subspace clustering, respectively. For details, refer to those paper [8, 9].

Recently, clustering on the subset of conditions has received significant attentions during the past few years [1, 12, 13, 15]. Many approaches first identify clusters in low dimensions (C) and derive clusters high dimensions based on



(a) Expression level of genes in subspace cluster 453

(b) Subspace cluster 453

(c) Expression level of genes in subspace cluster 625

(d) Subspace cluster 625

(e) Expression level of genes in subspace cluster 392

(f) Subspace cluster 392

Fig. 7. Sample plots for subspace clusters

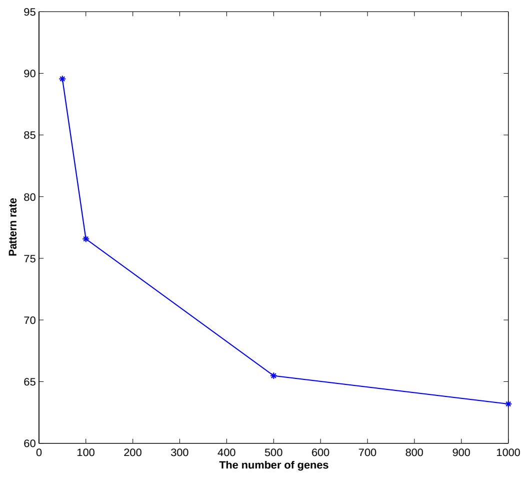


Fig. 8. The number of genes versus pattern rate

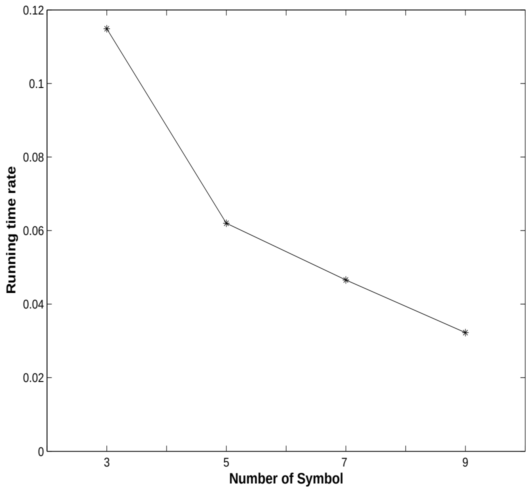


Fig. 9. Different number of symbols versus running time rate

C using the *apriori* principle. However, our method is different from other approaches in that all maximal subspaces for gene pairs (whose maximum value is bounded by $n(n-1)/2$) are generated by transforming a set of genes into a set of gene-gene relations. Identification of meaningful subspace clusters is based on the maximal patterns for gene pairs.

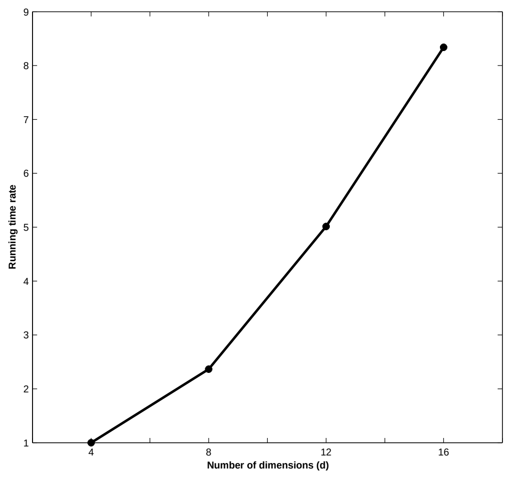


Fig. 10. The number of dimensions versus running time rate with the use of inverted index

5 Conclusion and Future Work

We presented the subspace mining framework that is vital to microarray data analysis. An experimental prototype system has been developed, implemented, and tested to demonstrate the effectiveness of the proposed algorithm. The uniqueness of our approach is based on observation that the maximum number of subspaces is limited to the number of genes. By transforming a gene-gene relation into a maximal pattern that is represented as a sequence of symbols, we could identify subspace clusters very efficiently by utilizing inverted index and pruning non-interesting subspaces. Experimental results indicated that our method is scalable to the number of dimensions.

We intend to extend this work into the following three directions. First, we are currently investigating efficient discretization algorithm for gene expression data. Second, we plan to conduct more comprehensive experiments on diverse microarray datasets as well as compare our approach with other competitive algorithms. Finally, in order to interpret obtained gene clusters, external knowledge needs to be involved. Toward this end, we plan to explore the methodology to quantify relationship between subspace clusters and known biology knowledge by utilizing Gene Ontology [2].

6 Acknowledgement

We would like to thank Dr. A. Fazel Famili at National Research Council of Canada for providing Cho's data. This research has been funded in part by the

Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152.

References

1. R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1998.
2. The Gene Ontology Consortium. Creating the gene ontology resource: design and implementation. *Genome Research*, 11(8):1425-1433, 2001.
3. R.J. Cho, M.J. Campbell, E.A. Winzeler,L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockhart, and R.W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:5-73, 1998.
4. S. Chung, J. Jun, and D. McLeod. Mining gene expression datasets using density-based clustering. In *Proceedings of ACM CIKM International Conference on Information and Knowledge Management*, 2004.
5. R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification (2nd Ed.).* Wiley, New York, 2001.
6. A. Gasch, and M. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3(11):1-22, 2002.
7. T.R. Golub *et al.* Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(15):531-537, 1999.
8. D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: a survey. In *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370-1386, 2004.
9. L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *ACM SIGKDD Explorations Newsletter*, 6(1):90-105, 2004.
10. G. Salton and M.J. McGill. *Introduction to modern information retrieval.* McGraw-Hill, 1983.
11. P. Tamayo *et al.* Interpreting patterns of gene expression with self organizing maps. In *Proceedings of National Academy of Science*, 96(6):2907-2912, 1999.
12. C. Tang, A. Zhang, and J. Pei. Mining phenotypes and informative genes from gene expression data. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.
13. H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2002.
14. Y. Xu, V. Olman, and D. Xu. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics*, 18(4):536-545, 2002.
15. M.J. Zaki, and M. Peters. CLICKS: Mining subspace clusters in categorical data via K-partite maximal cliques. In *Proceedings of International Conference on Data Engineering*, 2005.