TRICLUSTER: An Effective Algorithm for Mining Coherent Clusters in 3D Microarray Data

Lizhuang Zhao
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York, 12180
zhaol2@cs.rpi.edu

Mohammed J. Zaki
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York, 12180
zaki@cs.rpi.edu

ABSTRACT

In this paper we introduce a novel algorithm called TRICLUSTER, for mining coherent clusters in three-dimensional (3D) gene expression datasets. TRICLUSTER can mine arbitrarily positioned and overlapping clusters, and depending on different parameter values, it can mine different types of clusters, including those with constant or similar values along each dimension, as well as scaling and shifting expression patterns. TRICLUSTER relies on graph-based approach to mine all valid clusters. For each time slice, i.e., a gene×sample matrix, it constructs the range multigraph, a compact representation of all similar value ranges between any two sample columns. It then searches for constrained maximal cliques in this multigraph to yield the set of biclusters for this time slice. Then TRICLUSTER constructs another graph using the biclusters (as vertices) from each time slice; mining cliques from this graph yields the final set of triclusters. Optionally, TRICLUSTER merges/deletes some clusters having large overlaps. We present a useful set of metrics to evaluate the clustering quality, and we show that TRICLUSTER can find significant triclusters in the real microarray datasets.

1. INTRODUCTION

Traditional clustering algorithms work in the full dimensional space, which consider the value of each point in all the dimensions and try to group the similar points together. Biclustering [7], however, does not have such a strict requirement. If some points are similar in several dimensions (a subspace), they will be clustered together in that subspace. This is very useful, especially for clustering in a high dimensional space where often only some dimensions are meaningful for some subset of points. Biclustering has proved of great value for finding the interesting patterns in the microarray expression data [8], which records the expression levels of many genes (the rows/points), for different biological samples (the columns/dimensions). Biclustering is able to identify the co-expression patterns of a subset of genes that might be relevant to a subset of the samples of interest.

Besides biclustering along the gene-sample dimensions, there has been a lot of interest in mining gene expression patterns across time [4]. The proposed approaches are also mainly two-dimensional, i.e., finding patterns along the gene-time dimensions. In this paper, we are interested in mining tri-clusters, i.e., mining coherent clusters along the gene-sample-time (temporal) or gene-sample-region (spatial) dimensions. To the best of our knowledge, TRICLUSTER is *the first 3D microarray subspace clustering method.*

There are several challenges while mining microarray data for bi- and tri-clusters. First, biclustering itself is known to be a NP-hard problem [7], and thus many proposed algorithms of mining bicusters use heuristic methods or probabilistic approximations, which as a tradeoff decrease the accuracy of the final clustering results. Extending these methods to triclustering will be even harder. Second, microarray data is inherently susceptible to noise, due to varying experimental conditions, thus it is essential that the methods be robust to noise. Third, given that we do not understand the complex gene regulation circuitry in the cell, it is important that clustering methods allow overlapping clusters that share subsets of genes, samples or time-courses/spatial-regions. Furthermore, the methods should be flexible enough to mine several (interesting) types of clusters, and should not be too sensitive to input parameters.

In this paper, we present a novel, efficient, deterministic, triclustering method called TRICLUSTER, that addresses the above challenges. The key features of our approach include: 1) We mine only the maximal triclusters satisfying certain homogeneity criteria. 2) The clusters can be arbitrarily positioned anywhere in the input data matrix and they can have arbitrary overlapping regions. 3) we use a flexible definition of a cluster which can mine several types of triclusters, such as triclusters having identical or approximately identical values for all dimensions or a subset of the dimensions, and triclusters that exhibit a scaling or shifting expression values (where one dimension is a approximately constant multiple of or is at an approximately constant offset from another dimension, respectively). 4) TRICLUSTER is a deterministic and complete algorithm, which utilizes the inherent unbalanced property (number of genes being a lot more than the number of samples or time-slices) in microarray data, for efficient mining. 5) TRICLUSTER can optionally

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA.

merge/delete triclusters that have large overlaps, and can also automatically relax the similarity criteria. It can thus tolerate some noise in the dataset, and lets the user focus on the most important clusters. 6) We present a useful set of metrics to evaluate the clustering quality, and we show that triCluster can find substantially significant triclusters in the real microarray datasets.

2. PRELIMINARY CONCEPTS

Let $G = \{g_0, g_1, \cdots, g_{n-1}\}$ be a set of n genes, let $S = \{s_0, s_1, \cdots, s_{m-1}\}$ be a set of m biological samples (e.g., different tissues or experiments), and let $T = \{t_0, t_1, \cdots, t_{l-1}\}$ be a set of l experimental time points. A three dimensional microarray dataset is a real-valued $n \times m \times l$ matrix $D = G \times S \times T = \{d_{ijk}\}$ (with $i \in [0, n-1], j \in [0, m-1], k \in [0, l-1]$), whose three dimensions correspond to genes, samples and times respectively (note that the 3rd dimension can also be a spatial region of interest, but without loss of generality (w.l.o.g.), we will consider time as the 3rd dimension). Each entry d_{ijk} records the (absolute or relative) expression level of gene g_i in sample s_j at time t_k. For example, Table 1(a) shows a dataset with 10 genes, 7 samples and 2 time points. For clarity certain cells have been left blank; we assume that these are filled by some random expression values.

A *tricluster* C is a submatrix of the dataset D, where $C = X \times Y \times Z = \{c_{ijk}\}$, with $X \subseteq G$, $Y \subseteq S$ and $Z \subseteq T$ provided certain conditions of homogeneity are satisfied. For example, a simple condition might be that all values $\{c_{ijk}\}$ are identical or approximately equal. If we are interested in finding common gene co-expression patterns across different samples and times, we can find clusters that have similar values in the G dimension, but can have different values in the S and T dimensions. Other homogeneity conditions can also be defined, such as similar values in S dimension, order preserving submatrix, and so on [16].

Let $C = X \times Y \times Z = \{c_{ijk}\}$ be a tricluster and let $C_{2,2} = \begin{bmatrix} c_{ia} & c_{ib} \\ c_{ja} & c_{jb} \end{bmatrix}$ be any arbitrary 2×2 submatrix of C, i.e., $C_{2,2} \subseteq X \times Y$ (for some $z \in Z$) or $C_{2,2} \subseteq X \times Z$(for some $y \in Y$) or $C_{2,2} \subseteq Y \times Z$(for some $x \in X$). We call C a *scaling cluster* if we have $c_{ib} = \alpha_i c_{ia}$ and $c_{jb} = \alpha_j c_{ja}$, and further $|\alpha_i - \alpha_j| \leq \varepsilon$, i.e., the expression values differ by an approximately (within ε) constant *multiplicative factor* α. We call C a *shifting cluster* iff we have $c_{ib} = \beta_i + c_{ia}$ and $c_{jb} = \beta_j + c_{ja}$, and further $|\beta_i - \beta_j| \leq \varepsilon$, i.e., the expression values differ by a approximately (within ε) constant *additive factor* β.

We say that cluster $C = X \times Y \times Z$ is a subset of $C' = X' \times Y' \times Z'$, iff $X \subseteq X'$, $Y \subseteq Y'$ and $Z \subseteq Z'$. Let $\mathcal{B}$ be the set of all triclusters that satisfy the given homogeneity conditions, then $C \in \mathcal{B}$ is called a *maximal tricluster* iff there doesn't exist another cluster $C' \in \mathcal{B}$ such that $C \subset C'$ Let $C = X \times Y \times Z$ be a tricluster, and let $C_{2,2} = \begin{bmatrix} c_{ia} & c_{ib} \\ c_{ja} & c_{jb} \end{bmatrix}$ an arbitrary 2×2 submatrix of C, i.e., $C_{2,2} \subseteq X \times Y$ (for some $z \in Z$) or $C_{2,2} \subseteq X \times Z$(for some $y \in Y$) or $C_{2,2} \subseteq Y \times Z$(for some $x \in X$). We call C a **valid cluster** iff it is a maximal tricluster satisfying the following properties:

1. Let $r_i = |\frac{c_{ib}}{c_{ia}}|$ and $r_j = |\frac{c_{jb}}{c_{ja}}|$ be the ratio of two column values for a given row (i or j). We require that $\frac{\max(r_i, r_j)}{\min(r_i, r_j)} - 1 \leq \varepsilon$, where ε is a maximum ratio value.

Time t_0

	s_0	s_1	s_2	s_3	s_4	s_5	s_6
g_0	3.6	1.0	1.0		1.0	1.0	1.0
g_1	3.0	2.5			2.0		1.0
g_2		5.0			5.0		5.0
g_3	6.6	5.5					2.0
g_4	9.0	7.5			6.0		3.0
g_5	6.6				4.4		2.0
g_6		3.0			3.0		3.0
g_7		8.0	8.0		8.0	8.0	
g_8	6.0	5.0			4.0		2.0
g_9		4.0	4.0		4.0	4.0	4.0

Time t_1

	s_0	s_1	s_2	s_3	s_4	s_5	s_6
g_0		0.5	0.5		0.5	0.5	0.5
g_1		3.0			2.4		1.2
g_2		2.5			2.5		2.5
g_3		5.5					2.0
g_4		9.0			7.2		3.6
g_5					4.4		2.0
g_6		1.5			1.5		1.5
g_7		4.0	4.0		4.0	4.0	
g_8		6.0			4.8		2.4
g_9		2.0	2.0		2.0	2.0	2.0

(a)

Time t_0

	s_3	s_0	s_6	s_4	s_1	s_5	s_2
g_5		6.6	2.0	4.4			
g_2			5.0	5.0	5.0		
g_6			3.0	3.0	3.0		
g_0		3.6	1.0	1.0	1.0	1.0	1.0
g_9			4.0	4.0	4.0	4.0	4.0
g_7				8.0	8.0	8.0	8.0
g_3		6.6	2.0		5.5		
g_4		9.0	3.0	6.0	7.5		
g_8		6.0	2.0	4.0	5.0		
g_1		3.0	1.0	2.0	2.5		

Time t_1

	s_3	s_0	s_6	s_4	s_1	s_5	s_2
g_5							
g_2			2.5	2.5	2.5		
g_6			1.5	1.5	1.5		
g_0			0.5	0.5	0.5	0.5	0.5
g_9			2.0	2.0	2.0	2.0	2.0
g_7				4.0	4.0	4.0	4.0
g_3							
g_4			3.6	7.2	9.0		
g_8			2.4	4.8	6.0		
g_1			1.2	2.4	3.0		

(b)

Table 1: (a) Example Microarray Dataset. (b) Some Clusters

2. If $c_{ia} \times c_{ib} < 0$ then sign(c_{ia}) = sign(c_{ja}) and sign(c_{ib}) = sign(c_{jb}), where sign(x) returns -1/1 if x is negative/non-negative (expression values of zero are replaced with a small random positive correction value, in the preprocessing step). This allows us to easily mine datasets having negative expression values. [1]

[1]It also prevents us from reporting that, for example, expression ratio $-5/5$ is equal to $5/-5$.

3. We require that the cluster satisfy maximum range thresholds along each dimension. For any $c_{i_1 j_1 k_1} \in C$ and $c_{i_2 j_2 k_2} \in C$, let $\delta = |c_{i_1 j_1 k_1} - c_{i_2 j_2 k_2}|$. We require the following conditions: a) If $j_1 = j_2$ and $k_1 = k_2$, then $\delta \leq \delta^x$, b) if $i_1 = i_2$ and $k_1 = k_2$, then $\delta \leq \delta^y$, and c) if $i_1 = i_2$ and $j_1 = j_2$, then $\delta \leq \delta^z$. where δ^x, δ^y and δ^z represent the maximum range of expression values allowed along the gene, sample, and time dimensions.

4. We require that $|X| \geq mx$, $|Y| \geq my$ and $|Z| \geq mz$, where mx, my and mz denote minimum cardinality thresholds for each dimension.

LEMMA 1. *(Symmetry Property) Let $C = X \times Y \times Z$ be a tricluster, and let $\begin{bmatrix} c_{ia} & c_{ib} \\ c_{ja} & c_{jb} \end{bmatrix}$ an arbitrary 2×2 submatrix of $X \times Y$ (for some $z \in Z$) or $X \times Z$ (for some $y \in Y$) or $Y \times Z$ (for some $x \in X$). Let $r_i = |\frac{c_{ib}}{c_{ia}}|$, $r_j = |\frac{c_{jb}}{c_{ja}}|$, $r_a = |\frac{c_{ja}}{c_{ia}}|$, and $r_b = |\frac{c_{jb}}{c_{ib}}|$ then $\frac{\max(r_i,r_j)}{\min(r_i,r_j)} - 1 \leq \varepsilon \iff \frac{\max(r_a,r_b)}{\min(r_a,r_b)} - 1 \leq \varepsilon$.*

Proof: W.l.o.g. assume that $r_i \geq r_j$, then $|\frac{c_{ib}}{c_{ia}}| \geq |\frac{c_{jb}}{c_{ja}}| \iff |\frac{c_{ja}}{c_{ia}}| \geq |\frac{c_{jb}}{c_{ib}}| \iff r_a \geq r_b$. We now have $\frac{\max(r_i,r_j)}{\min(r_i,r_j)} = \frac{r_i}{r_j} = \frac{|c_{ib}/c_{ia}|}{|c_{jb}/c_{ja}|} = \frac{|c_{ja}/c_{ia}|}{|c_{jb}/c_{ib}|} = \frac{r_a}{r_b} = \frac{\max(r_a,r_b)}{\min(r_a,r_b)}$. Thus $\frac{\max(r_i,r_j)}{\min(r_i,r_j)} - 1 \leq \varepsilon \iff \frac{\max(r_a,r_b)}{\min(r_a,r_b)} - 1 \leq \varepsilon$. ■

The symmetric property of our cluster definition allows for very efficient cluster mining. The reason is that we are now free to mine clusters searching over the dimensions with the least cardinality. For example, instead of searching for subspace clusters over subsets of the genes (which can be large), we can search over subsets of samples (which are typically very few) or over subsets of time-courses (which are also not large).

Note that by definition, a cluster represents a scaling cluster (if the ratio is 1.0, then it is a uniform cluster). However, our our definition allows for the mining of shifting clusters as well, as indicated by the lemma below.

LEMMA 2. *(Shifting Cluster) Let $C = X \times Y \times Z = \{c_{xyz}\}$ be a maximal tricluster. Let $e^C = \{e^{c_{xyz}}\}$ be the tricluster obtained by applying the exponential function (base e) to each value in C. If e^C is a (scaling) cluster, then C is a shifting cluster.*

Proof: Let $C_{2,2} = \begin{bmatrix} c_{ia} & c_{ib} \\ c_{ja} & c_{jb} \end{bmatrix}$ an arbitrary 2×2 submatrix of C. Assume that e^C is a valid scaling cluster. Then by definition, $e^{c_{ib}} = \alpha_i e^{c_{ia}}$. But this immediately implies that $\ln(e^{c_{ib}}) = \ln(\alpha_i e^{c_{ia}})$, which gives us $c_{ib} = \ln(\alpha_i) + c_{ia}$. Likewise, we have $c_{jb} = \ln(\alpha_j) + c_{ja}$. Setting $\beta_i = \ln(\alpha_i)$ and $\beta_j = \ln(\alpha_j)$, we have that C is a shifting cluster. ■

Note that the clusters can have arbitrary positions anywhere in the data matrix, and they can have arbitrary overlapping regions (though, TRICLUSTER can optionally merge or delete overlapping clusters under certain scenarios). We impose the minimum size constraints i.e. mx, my and mz to mine large enough clusters. Typically $\varepsilon \approx 0$, so that the ratios of the values along one dimension in the cluster are similar (by symmetry Lemma 1, this property is also applicable for the other two dimensions), i.e., the ratios can differ by at most ε. Further, different choices of dimensional range thresholds (δ^x, δ^y and δ^z) produce different kinds of clusters:

a) If $\delta^x = \delta^y = \delta^z = 0$, then we obtain a cluster that has identical values along all dimensions.

b) If $\delta^x = \delta^y = \delta^z \approx 0$, then we obtain clusters with approximately identical values.

c) If $\delta^x \approx 0$, $\delta^y \neq 0$ and $\delta^z \neq 0$, then we obtain a cluster ($X \times Y \times Z$), where each gene $g_i \in X$ has similar expression values across the different samples Y and the different times Z, and different genes' expression values cannot differ by more than the threshold δ^x. Similarly we can obtain other cases by setting i) $\delta^x \neq 0$, $\delta^y \approx 0$ and $\delta^z \neq 0$ or ii) $\delta^x \neq 0$, $\delta^y \neq 0$ and $\delta^z \approx 0$.

d) $\delta^x \approx 0$, $\delta^y \approx 0$ and $\delta^z \neq 0$, we obtain a cluster with similar values for genes and samples, but the time-courses are allowed to differ by some arbitrary scaling factor. Similar cases are obtained by setting i) $\delta^x \approx 0$, $\delta^y \neq 0$ and $\delta^z \approx 0$, and ii) $\delta^x \neq 0$, $\delta^y \approx 0$ and $\delta^z \approx 0$.

e) If $\delta^x \neq 0$, $\delta^y \neq 0$ and $\delta^z \neq 0$, then we obtain a cluster that exhibits scaling behavior on genes, samples and times, and the expression values are bounded by δ^x, δ^y and δ^z respectively.

Note also that TRICLUSTER also allows different ε values for different pairs of dimensions. For example, we may use one value of ε to constrain the expression values for, say, the gene-sample slice, but we may then relax the maximum ratio threshold for the temporal dimension to capture more interesting (and big) changes in expression as time progresses.

For example, Table 1(b) shows some examples of different clusters that can be obtained by permuting some dimensions. Let $mx = my = 3$, $mz = 2$ and let $\varepsilon = 0.01$. If we let $\delta^x = \delta^y = \delta^z = \infty$, i.e., if they are unconstrained, then $C_1 = \{g_1, g_4, g_8\} \times \{s_1, s_4, s_6\} \times \{t_0, t_1\}$ is an example of a scaling cluster, i.e., each point values along one dimension is some scalar multiple of another point values along the same dimension. We also discover two other maximal overlapping clusters, $C_2 = \{g_0, g_2, g_6, g_9\} \times \{s_1, s_4, s_6\} \times \{t_0, t_1\}$, $C_3 = \{g_0, g_7, g_9\} \times \{s_1, s_2, s_4, s_5\} \times \{t_0, t_1\}$. Note that if we set $my = 2$ we would find another maximal cluster $C_4 = \{g_0, g_2, g_6, g_7, g_9\} \times \{s_1, s_4\} \times \{t_0, t_1\}$, which is subsumed by C_2 and C_3. We shall see later that TRICLUSTER can optionally delete such a cluster in the final steps. If we set $\delta^x = 0$, and let δ^y and δ^z be unconstrained, then we will not find cluster C_1, whereas all other clusters will remain valid. This is because if $\delta^x = 0$, then the values for each gene in the cluster must be identical, however since δ^y and δ^z are unconstrained the cluster can have different coherent values along the samples and times. Since ε is symmetric for each dimension, TRICLUSTER first discovers all unconstrained clusters rapidly, and then prunes unwanted clusters if δ^x, δ^y or δ^z are constrained. Finally, it optionally deletes or merges mined clusters if certain overlapping criteria are met.

3. RELATED WORK

While there has been work on mining gene expression patterns across time, to the best of our knowledge there is no previous method that mines tri-clusters. On the other hand, there are many full-space and biclustering algorithms designed to work with microarray datasets, such as feature based clustering [1, 2, 27], graph based clustering [13, 28,

24], and pattern based clustering [7, 15, 26, 18, 5]. Below, we briefly review some of these methods, and refer the reader to an excellent, recent survey on biclustering [16] for more details. We first begin by discussing time-series expression clustering methods.

3.1 Time-based Microarray Clustering

Jiang et al [14] gave a method to analyze the gene-sample-time microarray data. It treats the gene-sample-time microarray data as a gene×sample matrix with each entry as a vector of the values along the time dimension. For any two such vectors, it uses their *Pearson's correlation coefficient* as the distance. Then for each gene, it groups similar time-vectors together to form a sample subset. After that, it enumerates the subset of all the genes to find those subsets of genes whose corresponding sample subsets result in a considerable intersection set. The paper discussed two methods: grouping samples first and grouping genes first. Although the paper dealt with three dimensional microarray data, it considers the time dimension in full space (i.e. all the values along the time dimension), and is thus unable to find temporal trends that are applicable to only a subset of the times, and as such it casts the 3D problem into a biclustering problem.

In general, most previous methods apply traditional full space clustering (with some improvements) to the gene time-series data. Thus these methods are not capable of mining coherent subspace clusters, i.e., these methods sometimes will miss important information obscured by the data noise. For example, Erdal et al. [9] extract a 0/1 vector for each gene, such that there is a '1' whenever there is a big change in its expression from one time to the next. Using longest common subsequence length as similarity, they perform a full-dimensional clustering. The subspaces of time-points are not considered, and the sample space is ignored. Moller et al [17] present another time-series microarray clustering algorithm. For any two time vectors $[x_1(t_1), x_2(t_2), \cdots, x_k(t_k)]$ and $[y_1(t_1), y_2(t_2), \cdots y_k(t_k)]$ they calculate $sim(x, y) = \sum_{k=1}^{n} \frac{(x_{k+1}-x_k)-(y_{k+1}-y_k)}{t_{k+1}-t_k}$. Then they use a full-space repeated fuzzy clustering algorithm to partition the time-series clusters. Ramoni et al [20] presents a Bayesian method for model-based gene expression clustering. It represents gene expression dynamics as autoregressive equations and uses an agglomerative method to search for the clusters. Feng et al [10] proposed a time-frequency based full-space algorithm using a measure of functional correlation set between time-course vectors of different genes. Filkov et al [11] addressed the analysis of short-term time-series gene microarray data, by detecting the period in a predominantly cycling dataset, and the phase between phase-shifted cyclic datasets. It too is a fullspace clustering method on gene time-series data. For a more comprehensive look at time-series gene expression analysis, see the recent paper by Bar-Joseph [4]. The paper divides the computational challenges into four analysis levels: experimental design, analysis, pattern recognition and gene networks. It discusses the computational and biological problems at each level and reviews some methods proposed to deal with these issues. It also highlights some open problems.

3.2 Feature- and Graph-based Clustering

PROCLUS[1] and ORCLUS [2] use projective clustering to partition the dataset into clusters occurring in possibly different subsets of dimensions in a high dimensional dataset. PROCLUS seeks to find axis-aligned subspaces by partitioning the set of points and then uses a hill-climbing technique to refine the partitions. ORCLUS finds arbitrarily oriented clusters by using ideas related to singular value decomposition. Other subspace clustering methods include CLIQUE [3] and DOC [19]. These methods are not designed to mine coherent patterns from microarray datasets.

CLIFF [27] iterates between feature (genes) filtering and sample partitioning. It first calculates k best features (genes) according to their intrinsic discriminability using current partitions. Then it partitions the samples with these features by keeping the minimum normalized weights. This process iterates until convergence. COSA [12] allows traditional clustering algorithms to cluster on a subset of attributes, rather than all of them. Principal Component Analysis for gene expression clustering has also been proposed [30].

HCS [13] is a fullspace clustering algorithm. It cuts a graph into subgraphs by removing some edges, and repeats until all the vertices in each subgraph are similar enough. MST [28] is also a fullspace clustering method. It uses greedy method to construct a minimum spanning tree, and splits the current tree(s) repeatedly, until the average edges length in each subtree is below some threshold. Then each tree is a cluster. SAMBA [24] uses a bipartite graph to model and implement the clustering. It repeatedly finds the maximal highly-connected sub-graph in the bipartite graph. Then it performs local improvement by adding or deleting a single vertex until no further improvement is possible. Other graph-theoretic clustering algorithms include CLICK [21] and CAST [6].

There are some common drawbacks concerning the above algorithms applied to microarray datasets. First, some of them are randomized methods based on shrinking and expansion, which sometimes results in incomplete clusters. Second, none of them can deal with overlapped clusters properly. Third, the greedy methods will lead to a local optimum that may miss some important (part of) clusters. Moreover, fullspace clustering is even not biclustering and will compromise the important clusters by considering irrelevant dimensions. In general, none of them are deterministic, and thus cannot guarantee that all valid (overlapped) clusters are found.

3.3 Pattern-based Clustering

δ-biclustering [7] uses mean squared residue of a submatrix ($X \times Y$) to find biclusters. If a submatrix with enough size has a mean squared residue less than threshold δ, it is a δ-bicluster. Initially, the algorithm starts with the whole data matrix, and repeatedly adds/deletes a row/column from the current matrix in a greedy way until convergence. After having found a cluster, it replaces the submatrix with random values, and continues to find the next best cluster. This process iterates until no further clusters can be found. One limitation of δ-biclustering is that it may converge to a local optimum, and cannot guarantee all clusters will be found. Also it can easily miss overlapping clusters due to the random value substitutions it does. Another move-based δ-biclustering method was proposed in [29]. However, it too is an iterative improvement based method.

pCluster [26] defines a cluster C as a submatrix of the original dataset, such that for any two by two submatrix

$\begin{bmatrix} c_{xa} & c_{xb} \\ c_{ya} & c_{yb} \end{bmatrix}$ of C, $|(c_{xa} - c_{ya}) - (c_{xb} - c_{yb})| < \delta$, where δ is a threshold. The algorithm first scans the dataset to find all column-pair and row-pair maximal clusters called MDS. Then it does the pruning in turn using the row-pair MDS and the column-pair MDS. It then mines the final clusters based on a prefix tree. pCluster is symmetric, i.e., it treats rows and columns equally, and it is capable of finding similar clusters as triCluster, but it does not merge/prune clusters and is not robust to noise. Further, we show that it runs much slower than triCluster on real microarray datasets.

xMotif [18] requires all genes expressions in a bicluster to be similar across all the samples. It randomly picks a seed sample s and a sample subset d (called discriminating set), and then finds all such genes that are conserved across all the samples in d. xMotif uses Monte Carlo method to find the clusters that cover all genes and samples. However it cannot guarantee to find all the clusters because of its random sampling process.

Another stochastic algorithm OPSM [5] has similar drawback as xMotif, but uses a different cluster definition. It defines a cluster as a submatrix of the original data matrix after row and column permutation, whose row values are in a non-decreasing pattern. Another method using this definition is OP-Cluster [15]. Gene clustering methods using Self Organizing Maps [23], and iterated two-way clustering [25] have also been proposed; a systematic comparison these and other biclustering methods can be found in [16].

4. THE triCluster ALGORITHM

As outlined above, triCluster mines arbitrarily positioned and overlapping scaling and shifting patterns from a three dimensional dataset, as well as several specializations. Typically 3D microarray datasets have more genes than samples, and perhaps an equal number of time points and samples, i.e., $|G| \geq |T| \approx |S|$. Due to the symmetric property, triCluster always transposes the input 3D matrix such that the dimension with the largest cardinality (say G) is 1st dimension; we then make S as the 2nd and T as the 3rd dimension. triCluster has three main steps: 1) For each $G \times S$ time slice matrix, find the valid ratio-ranges for all pair of samples, and construct a range multigraph, 2) Mine the maximal biclusters from the range multigraph, 3) Construct a graph based on the mined biclusters (as vertices) and get the maximal triclusters, and 4) Optionally, delete or merge clusters if certain overlapping criteria are met. We look at each step below.

4.1 Construct Range Multigraph

Given a dataset D, the minimum size thresholds, mx, my and mz, and the maximum ratio threshold ε, let s_a and s_b be any two sample columns in some time t of D and let $r_x^{ab} = \frac{d_{xa}}{d_{xb}}$ be the ratio of the expression values of gene g_x in columns s_a and s_b, where $x \in [0, n-1]$. A *ratio range* is defined as an interval of ratio values, $[r_l, r_u]$, with $r_l \leq r_u$. Let $\mathcal{G}_{ab}([r_l, r_u]) = \{g_x : r_x^{ab} \in [r_l, r_u]\}$ be the set of genes, called the *gene-set*, whose ratios w.r.t. columns s_a and s_b lie in the given ratio range, and if $r_x^{ab} < 0$ all the values in the same column have same signs (negative/non-negative).

In the first step triCluster quickly tires to summarize the valid ratio ranges that can contribute to some bicluster. More formally, we call a ratio range *valid* iff: 1) $\frac{max(|r_u|,|r_l|)}{min(|r_u|,|r_l|)} - 1 \leq \varepsilon$, i.e., the range satisfies the maximum ratio threshold imposed in our cluster definition. 2) $|\mathcal{G}_{ab}([r_l, r_u])| \geq mx$, i.e., there are enough (at least mx) genes in the gene-set. This is imposed since our cluster definition requires any cluster to have at least mx genes. 3) If there exists a $r_x^{ab} < 0$, all the values $\{d_{xa}\}/\{d_{xb}\}$ in the same column have same signs (negative/non-negative). 4) $[r_l, r_u]$ is maximal w.r.t. ε, i.e., we cannot add another gene to $\mathcal{G}_{ab}([r_l, r_u])$ and yet preserve the ε bound. Intuitively, we want to find all the maximal ratio ranges that satisfy the ε threshold, and span at least mx genes. Note that there can be multiple ranges between two columns and also that some genes may not belong to any range.

Figure 1 shows the ratio values for different genes using columns s_0/s_6 at time t_0 for our running example in Table 1. Assume $\varepsilon = 0.01$, and $mx = 3$, then there is only one valid ratio range, $[3.0, 3.0]$ and the corresponding gene-set is $\mathcal{G}_{s_0 s_6}([3.0, 3.0]) = \{g_1, g_4, g_8\}$. Using a sliding window approach (with window size: $r_x^{06} \times \varepsilon$ for each gene g_x) over the sorted ratio values, triCluster find all valid ratio ranges for all pairs of columns $s_a, s_b \in S$. If at any stage there are more than mx rows within the window, a range is generated. It is clear that different ranges may overlap. For instance, if we let $\varepsilon = 0.1$ we would obtain two valid ranges, $[3.0, 3.3]$ and $[3.3, 3.6]$, with overlapping gene-sets $\{g_1, g_4, g_8, g_3, g_5\}$ and $\{g_3, g_5, g_0\}$ respectively. If there are consecutive overlapping valid ranges, we merge them into an extended range, even though the maximum ratio threshold ε is exceeded. If the extended range is too wide, say more than 2ε, we split the extended range into several blocks of range at most 2ε (*split ranges*). To avoid missing any potential clusters, we also add some overlapping *patched ranges*. This process is illustrated in Figure 1(b). Note that an added advantage of allowing extended ranges is that it makes the method more robust to noise, since often the users may set a stringent ε condition, whereas the data might require a larger value.

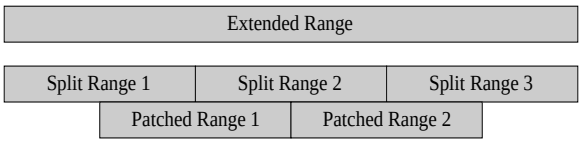
Given the set of all valid, as well as the extended split or patched ranges, across any pairs of columns s_a, s_b with $a < b$, given as $\mathcal{R}^{ab} = \{R_i^{ab} = [r_{l_i}^{ab}, r_{u_i}^{ab}] \ : \ s_a, s_b \in S\}$, we construct a weighted, directed, range multigraph $M = (V, E)$, where $V = S$ (the set of all samples), and for each $R_i^{ab} \in \mathcal{R}^{ab}$ there exists a weighted, directed edge $(s_a, s_b) \in E$ with weight $w = \frac{r_{u_i}^{ab}}{r_{l_i}^{ab}}$. In addition, each edge in the range multigraph has associated with it the gene-set corresponding to the range on that edge. For example, suppose $my = 3$, $mx = 3$, and $\varepsilon = 0.01$. Figure 2 shows the range multigraph constructed from Table 1, for time t_0. Another range multigraph is obtained for time t_1.

4.2 Mine Biclusters from Range Multigraph

The range multigraph represents in a compact way all the valid ranges that can be used to mine potential biclusters corresponding to each time slice, and thus filters out most of the unrelated data. biCluster uses a depth first search (DFS) on the range multigraph to mine all the biclusters, as shown in pseudo-code in Figure 3. It takes as input the set of parameter values $\varepsilon, mx, my, \delta^x, \delta^y$, the range graph M^t for a given time point t, and the set of genes all G and samples S. It will output the final set of all biclusters $\mathcal{C}^t$ for that time course. biCluster is a recursive algorithm, that at each call accepts a current *candidate* bicluster $C = X \times Y$, and a set of not yet expanded samples P. The initial call

s_0/s_6	3.0	3.0	3.0	3.3	3.3	3.6
Row	g_1	g_4	g_8	g_3	g_5	g_0

(a)



(b)

Figure 1: (a) Sorted ratios of column s_0/s_6 in table 1. (b) Split and patched ranges

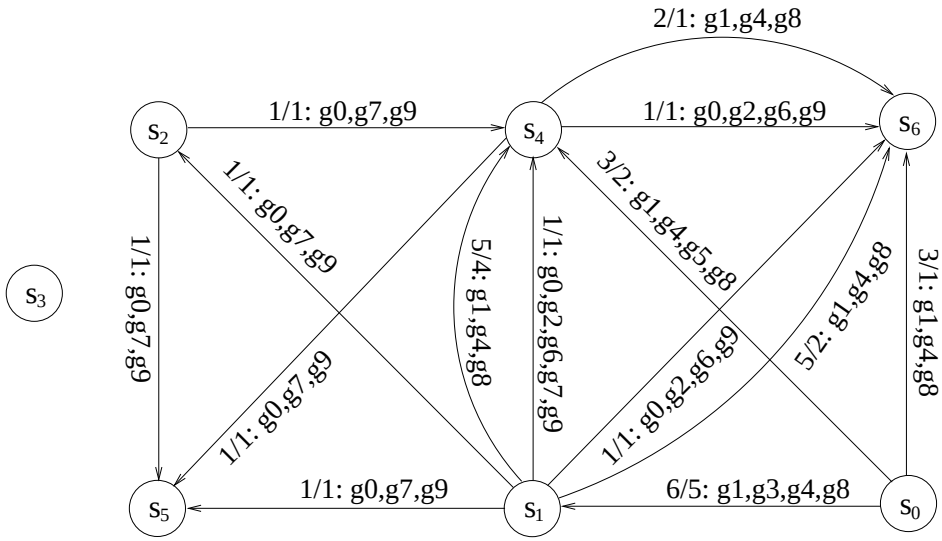


Figure 2: Weighted, directed Range multigraph

Input : parameters: $\varepsilon, mx, my, \delta^x, \delta^y$, range graph M_t, set of genes G and samples S
Output : bicluster set $\mathcal{C}^t$
Initialization: $\mathcal{C}^t = \emptyset$, call biCluster$(C = G \times \emptyset, S)$

```
1  biCluster (C = X×Y, P):
2  if C satisfies δ^x, δ^y then
3      if |C.Y| ≥ my then
4          if ∄C' ∈ C^t, such that C ⊂ C' then
5              Delete any C'' ∈ C, if C'' ⊂ C
6              C^t ⟵ C^t + C
7  foreach s_b ∈ P do
8      C^new ⟵ C
9      C^new.Y ⟵ C^new.Y + s_b
10     P ⟵ P − s_b
11     if C.Y = ∅ then
12         biCluster(C^new, P)
13     else
14         forall s_a ∈ C.Y and each R_i^ab ∈ R^ab
           satisfying |G(R_i^ab) ∩ C.X| ≥ mx do
15             C^new.X ⟵ (⋂_{all s_a∈C.Y} G(R_i^ab)) ∩ C.X
16             if |C^new.X| ≥ mx then
17                 biCluster(C^new, P)
```

Figure 3: biCluster Algorithm

is made with a cluster $C = G \times \emptyset$ with all genes G, but no samples, and with $P = S$, since we have not processed any samples yet. Before passing C to the recursive call we make sure that $|C.X| \geq mx$ (which is certainly true in the initial call, and also at line 16). Line 2 checks if the cluster meets the the maximum gene and sample range thresholds δ^x, δ^y, and also the minimum sample cardinality my (line 3). If so, we next check if C is not already contained in some maximal cluster $C' \in \mathcal{C}^t$ (line 3). If not, then we add C to the set of final clusters $\mathcal{C}^t$ (line 6), and we remove any cluster $C'' \in \mathcal{C}$ already subsumed by C (line 5).

Lines 7-17 generate a new candidate cluster by expanding the current candidate by one more sample, constructs the appropriate gene-set for the new candidate, before making a recursive call. biCluster begins by adding to the current cluster C, each new sample $s_b \in P$ (line 7), to obtain a new candidate C^{new} (lines 8-9). Samples already processed are removed from P (line 10). Let s_a be all samples added to C until previous recursive call. If no previous vertex s_a exists (which will happen during the initial call, when $C.Y = \emptyset$), then we simply call biCluster with the new candidate. Otherwise, biCluster tries all combinations of each qualified range edge R_i^{ab} between s_a and s_b for all $s_a \in C.Y$ (line 14), obtains their gene-set intersection $\bigcap_{all\ s_a \in C.Y} \mathcal{G}(R_i^{ab})$ and intersects with $C.X$ to obtain the valid genes in the new cluster C^{new} (line 15). If the new cluster has at least mx genes, then another recursive call to biCluster is made (lines 16-17).

For example, let's consider how the clusters are mined from the range graph M^{t_0} shown in Figure 2. Let $mx = 3, my = 3, \varepsilon = 0.01$ as before. Initially biCluster starts at vertex s_0 with the candidate cluster $\{g_0, \cdots, g_9\} \times \{s_0\}$.

We next process vertex s_1; since there is only one edge, we obtain a new candidate $\{g_1, g_3, g_4, g_8\} \times \{s_0, s_1\}$. From s_1 we process s_4 and consider both the edges: for $w = 5/4, \mathcal{G} = \{g_1, g_4, g_8\}$, we obtain the new candidate $\{g_1, g_4, g_8\} \times \{s_0, s_1, s_4\}$, but the other edge $w = 1/1, \mathcal{G} = \{g_0, g_2, g_6, g_7, g_9\}$ will not have enough genes. We then further process s_6. Out of the two edges between s_4 and s_6, only one (with weight 2/1) will yield a candidate cluster $\{g_1, g_4, g_8\} \times \{s_0, s_1, s_4, s_6\}$. Since this is maximal, and meets all parameters, at this point we have found one (C_1) of the three final clusters shown in Figure 1(b). Likewise, when we start from s_1, we will find the other two clusters $C_3 = \{g_0, g_7, g_9\} \times \{s_1, s_2, s_4, s_5\}$ and $C_2 = \{g_0, g_2, g_6, g_9\} \times \{s_1, s_4, s_6\}$. Intuitively, we are searching for maximal cliques (on samples), with cardinality at least my, that also satisfy the minimum number of genes constraint mx.

4.3 Get Triclusters from Bicluster Graph

Input : parameters: $\varepsilon, mx, my, mz, \delta^x, \delta^y, \delta^z$, bicluster sets $\{\mathcal{C}^t\}$ of all times, set of genes G, samples S and times T
Output : cluster set $\mathcal{C}$
Initialization: $\mathcal{C} = \emptyset$, call triCluster($C = G \times S \times \emptyset, T$)

```
1  triCluster (C = X×Y×Z, P);
2  if C satisfies δ^x, δ^y, δ^z then
3      if |C.Z| ≥ mz then
4          if ∄C' ∈ C, such that C ⊂ C' then
5              Delete any C'' ∈ C, if C'' ⊂ C
6              C ⟵ C + C
7  foreach t_b ∈ P do
8      C^new.Z ⟵ C.Z + t_b
9      P ⟵ P − t_b
10     forall t_a ∈ C.Z and each bicluster c_i^{t_a} ∈ C^{t_a}
       satisfying |c_i^{t_a}.X ∩ C.X| ≥ mx and
       |c_i^{t_a}.Y ∩ C.Y| ≥ my do
11         C^new.X ⟵ (⋂_{all t_a∈C.Z} c_i^{t_a}.X) ∩ C.X
12         C^new.Y ⟵ (⋂_{all t_a∈C.Z} c_i^{t_a}.Y) ∩ C.Y
13         if |C^new.X| ≥ mx and |C^new.Y| ≥ my and
           the ratios at time t_b, t_a are coherent then
14             triCluster(C^new, P)
```

Figure 4: triCluster Algorithm

After having got the maximal bicluster set $\mathcal{C}^t$ for each time slice t, we use them to mine the maximal triclusters. This is accomplished by enumerating the subsets of the time slices as shown in Figure 4, using a process similar to the biCluster clique mining (Figure 3). For example, from Table 1, we can get the biclusters from the two time points t_0 and t_1 as shown in Figure 5. Since the clusters are identical, to adequately illustrate our tricluster mining method, let's assume that we also obtain other biclusters at times points t_3 and t_8. Assume the minimum size threshold is $mx \times my \times mz = 3 \times 3 \times 3$. triCluster starts from time t_0, which contains three biclusters. Let's begin with cluster C_1 at time t_0, given as $C_1^{t_0}$. For each bicluster $\mathcal{C}^{t_1}$, only $C_1^{t_1}$ can be used for extension since $C_1^{t_0} \cap C_1^{t_1} = \{g_1, g_4, g_8\} \times \{s_0, s_1, s_4, s_6\}$,

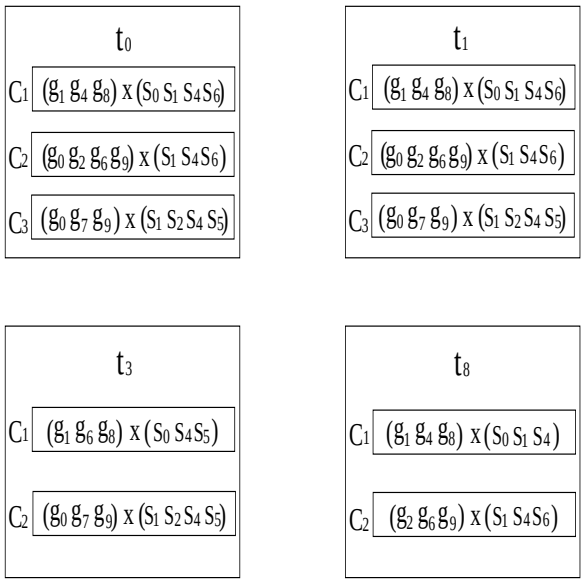


Figure 5: Tricluster example

which can satisfy the cardinality constraints (Figure 4, line 15). We continue by processing time t_3, but the cluster cannot be extended. So we try t_8, and we may find that we can extend it via $C_1^{t_8}$. The final result of this path is $\{g_1, g_4, g_8\} \times \{s_0, s_1, s_4, \} \times \{t_0, t_1, t_8\}$. Similarly we try all such paths and keep maximal triclusters only. During this process we also need to check the coherent property along the time dimension, as the tricluster definition requires, between the new time slice and the previous one. For example, for the three biclusters in Table 1, the ratios between t_1 and t_0 are 1.2 (for C_1) and 0.5 (for C_2 and C_3) respectively. If the extended bicluster has no such coherent values in the intersection region, triCluster will prune it.

The complexity of this part (along the time dimension) is the same as that of biclusters generation (biCluster) for one time slice. But since the biCluster need to run $|T|$ times, the total running time is $|T| \times (time(multigraph) + time(biCluster)) + time(triCluster)$.

4.4 Merge and Prune Clusters

After mining the set of all clusters, triCluster optionally merges or deletes certain clusters with large overlap. This is important, since real data can be noisy, and the users may not know the correct values for different parameters. Furthermore, many clusters having large overlaps only make it harder for the users to select the important ones.

Let $A = X_A \times Y_A \times Z_A$, and $B = X_B \times Y_B \times Z_B$ be any two mined clusters. We define the *span* of a cluster $C = X \times Y \times Z$, to be the set of gene-sample-time tuples that belong to the cluster, given as $L_C = \{(g_i, s_j, t_k) | g_i \in X, s_j \in Y, t_k \in Z\}$. Then we can define the following derived spans:

- $L_{A \cup B} = L_A \cup L_B$,
- $L_{A-B} = L_A - L_B$, and
- $L_{A+B} = L_{(X_A \cup X_B) \times (Y_A \cup Y_B) \times (Z_A \cup Z_B)}$

If any of the following three overlap conditions are met, triCluster either deletes or merges the clusters involved:

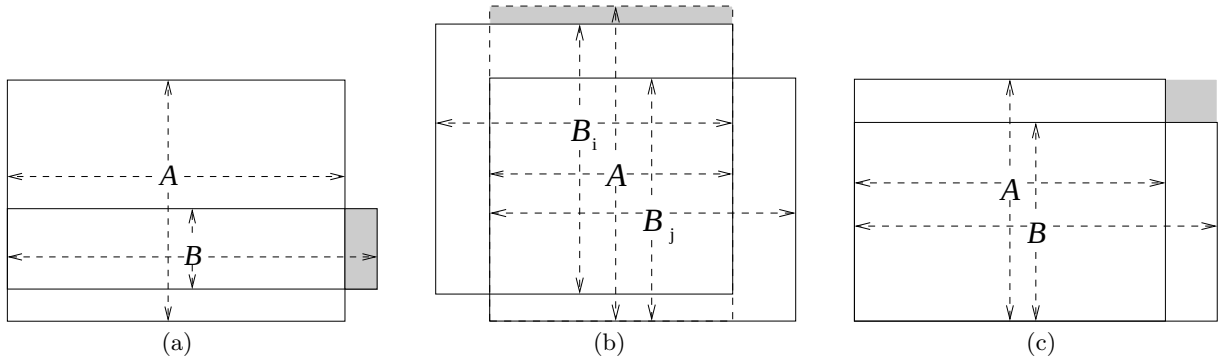


Figure 6: Three pruning or merging cases

1. 1) For any two clusters A and B, if $L_A > L_B$, and if $\frac{|L_{B-A}|}{|L_B|} < \eta$, then delete B. As illustrated in Figure 6(a) (for clarity, we use two dimensional figures here), this means that if the cluster with the smaller span (B) has only a few extra elements, then delete the smaller cluster.

2. 2) This is a generalization of case 1). For a cluster A, if there exists a set of clusters $\{B_i\}$, such that $\frac{|L_A - L_{\cup_i B_i}|}{|L_A|} < \eta$, then delete cluster A. As shown in Figure 6(b), A is mostly covered by the $\{B_i\}$'s. Therefore it can be deleted.

3. 3) For two clusters A and B, if $\frac{|L_{(A+B)-A-B}|}{|L_{A+B}|} < \gamma$, merge A and B into one cluster $(X_A \cup X_B) \times (Y_A \cup Y_B) \times (Z_A \cup Z_B)$. This case is shown in Figure 6(c). Here η, γ are user-defined thresholds.

4.5 Complexity Analysis

Since we have to evaluate all pairs of samples, compute their ratios, and find the valid ranges over all the genes, the range multigraph construction step takes time $O(|G||S|^2|T|)$. The bicluster mining step and tricluster mining step correspond to constrained maximal clique enumeration (i.e., cliques satisfying the $mx, my, mz, \delta^x, \delta^y, \delta^z$ parameters) from the range multigraph and bicluster graph. Since in the worst case there can be an exponential number of clusters, these two steps are the most expensive. The precise number of clusters mined depends on the dataset and the input parameters. Nevertheless, for microarray datasets triCluster is likely to be very efficient due to the following reasons: First, the range multigraph prunes away much of the noise and irrelevant information. Second, the depth of the search is likely to be small, since microarray datasets have far fewer samples and times than genes. Third, triCluster keeps intermediate gene-sets for all candidate clusters, which can prune the search the moment the input criteria are not met. The merging and pruning step apply only to those pairs of clusters that actually overlap, which can be determined in $O(|\mathcal{C}| \log(|\mathcal{C}|))$ time.

5. EXPERIMENTS

Unless otherwise noted, all experiments were done on a Linux/Fedora virtual machine (Pentium-M, 1.4GHz, 448M memory) over Windows XP through middleware VMware. We used both synthetic and real microarray datasets to evaluate triCluster algorithm. For the real dataset we used the yeast cell-cycle regulated genes [22] (http:// genome-www.stanford.edu/cellcycle). The goal is the study was to identify all genes whose mRNA levels are regulated by the cell cycle.

Synthetic datasets allow us to embed clusters, and then to test how triCluster performs for varying input parameters. We generate synthetic data using the following steps: The input parameters to the generator are the total number of genes, samples and times; number of clusters to embed; percentage of overlapping clusters; dimensional ranges for the cluster sizes; and the amount of noise for the expression values. The program randomly picks cluster positions in the data matrix, ensuring that no more than the required number of clusters overlap. The cluster sizes are generated uniformly between each dimensional ranges. For generating the expression values within a cluster, we generate at random, base values (v_i, v_j and v_k) for each dimension in the cluster. Then the expression value is set as $d_{ijk} = v_i \cdot v_j \cdot v_k \cdot (1 + \rho)$, where ρ doesn't exceed the random noise level. Once all clusters are generated, the non-cluster regions are assigned random values.

5.1 Results on Synthetic Datasets

We first wanted to see how triCluster behaves with varying input parameters in isolation. We generated synthetic data with the following default parameters: data matrix size $4000 \times 30 \times 20 (G \times S \times T)$, number of clusters 10, cluster size $150 \times 6 \times 4 (X \times Y \times Z)$, percentage overlap 20%, noise level 3%. For each experiment, we keep all default parameters, except for the varying parameter. We also choose appropriate parameter values for triCluster so that all embedded clusters were found. Figure7(a)-(f) shows triCluster's sensitivity to different parameters. We found that the time increases approximately linearly with the number of genes in a cluster (a). This is because, the range multi-graph is constructed on the samples, and not on the genes; more genes lead to longer gene-sets (per edge), but the intersection time is essentially linear in gene-set size. The time is exponential with the number of samples (b), since we search over the sample subset space. Finally, the time for increasing time-slices is also linear for the range shown (c), but in general the dependence will be exponential, since triCluster searches over subsets of time points after mining the biclusters for each time-slice. The time is linear w.r.t. number of clusters (d), whereas the overlap percentage doesn't seem to have much impact on the time (e). Finally, as we add more noise, the more the time to mine the clusters (f), since there is more chance that a random

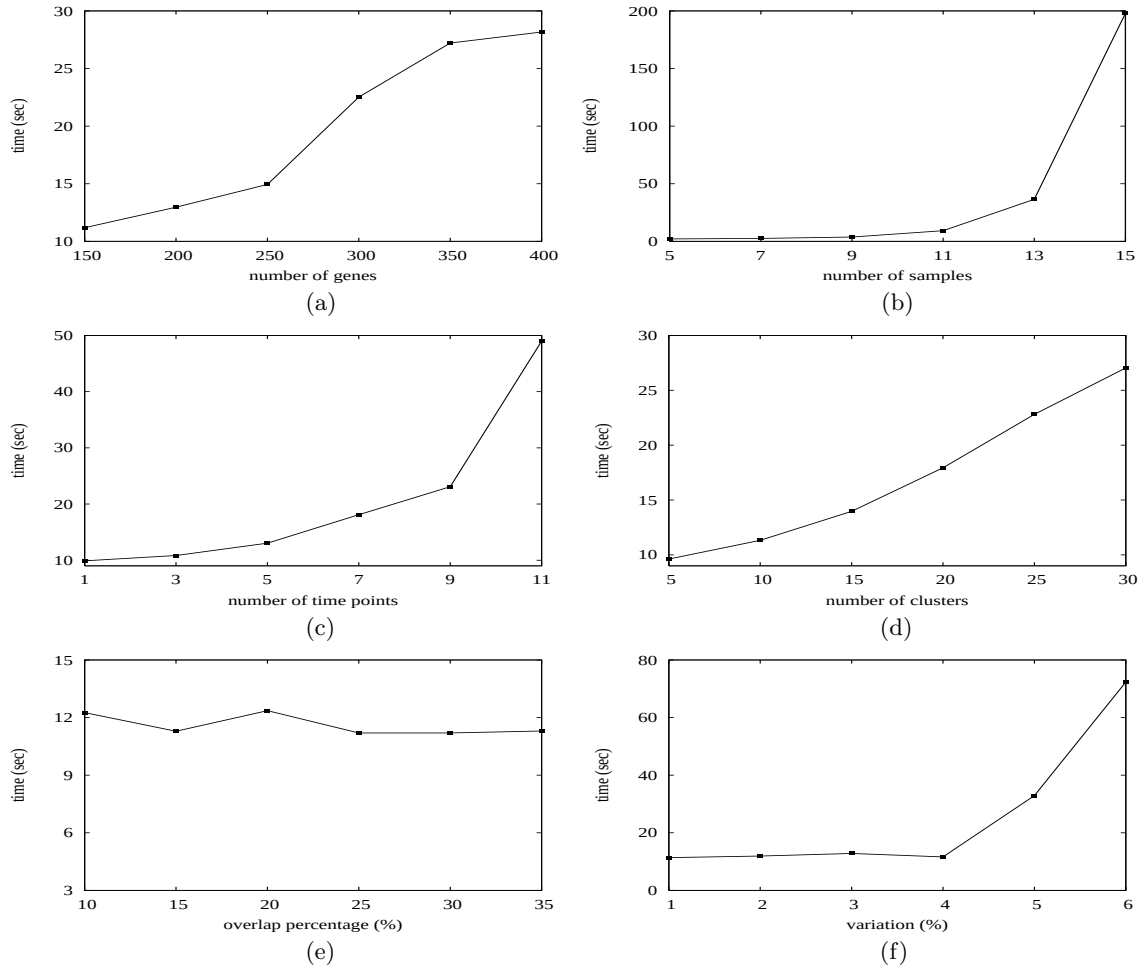


Figure 7: Evaluation of TRICLUSTER on Synthetic Datasets

gene or sample can belong to some cluster.

5.2 Results from Real Microarray Datasets

We define several metrics to analyze the output from different biclustering algorithms. If $\mathcal{C}$ is the set of all clusters output, then

1. *Cluster#* is just $|\mathcal{C}|$
2. *Element_Sum* is the sum of the spans of all clusters, i.e., $Element\_Sum = \sum_{C \in \mathcal{C}} |L_C|$
3. *Coverage* is the span of the union of all clusters, i.e., $Coverage = |L_{\bigcup_{C \in \mathcal{C}} C}|$
4. *Overlap* is given as $\frac{Element\_Sum - Coverage}{Coverage}$
5. *Fluctuation* is the average variance across a given dimension across all clusters.

For the yeast cell cycle data we looked at the time slices for the Elutritration experiments. There are a total of 7679 genes whose expression value is measured from time 0 to 390 minutes at 30 minute intervals. Thus there are 14 total time points. Finally, we use 13 of the attributes of the raw data as the samples (e.g., the raw values for the average and normalized signal for Cy5 & Cy3 dyes, the ratio of those values, etc.). Thus we obtain a 3D expression matrix of size: $T \times S \times G = 14 \times 13 \times 7679$. We mined this data looking for triclusters with minimum size at least $mx = 50$ (genes), $my = 4$ (samples), $mz = 5$ (time points), and we set $\varepsilon = 0.003$ (however, we relax the ε threshold along the time dimension). The per dimension thresholds $\delta^x, \delta^y, \delta^z$ were left unconstrained. TRICLUSTER output 5 clusters in 17.8s, with the following metrics:

Clusters#	5
Elements#	6520
Coverage	6520
Overlap	0.00%
Fluctuation	T:626.53, S:163.05, G:407.3

We can see that none of the 5 clusters was overlapping. The values for the total span across the clusters was 6520 cells, and the variances along each dimension are also shown. To visually see a mined tricluster, we plot various 2D views of one of the clusters (C_0) in Figure 8, Figure 9, and Figure 10. Figure 8 shows how the expression values for the genes (X axis) changes across the samples (Y axis), for different time points (the different sun-plots). Figure 9 shows how the gene expression (X axis) changes across the different time slices (Y axis), for different samples (the different sub-plots). Finally Figure 9 shows what happens at different times (X axis) for different genes (Y axis), across different samples (the different sub-plots). These figures show that TRICLUSTER is able to mine coherent clusters across any

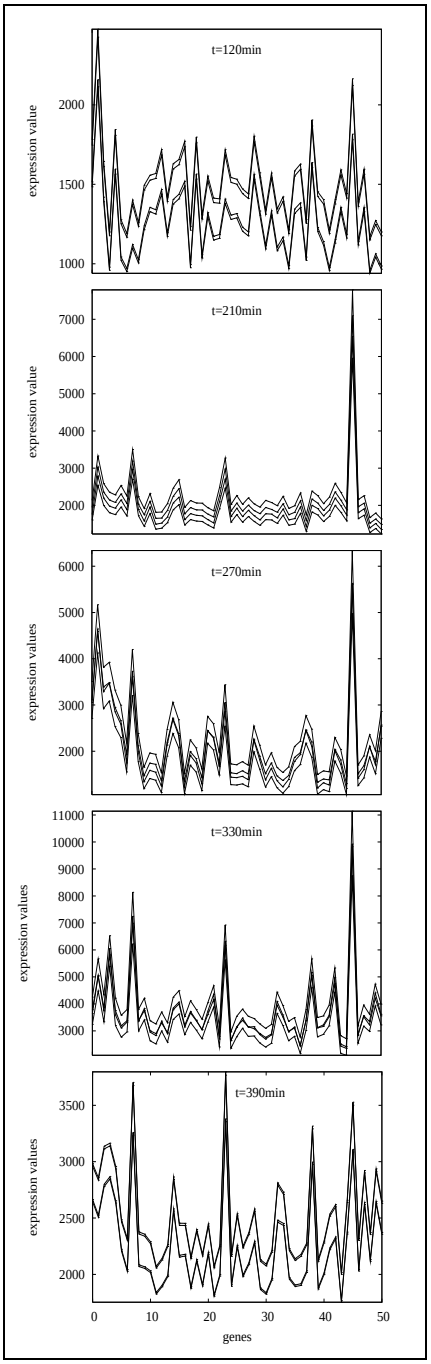


Figure 8: Sample-Curves

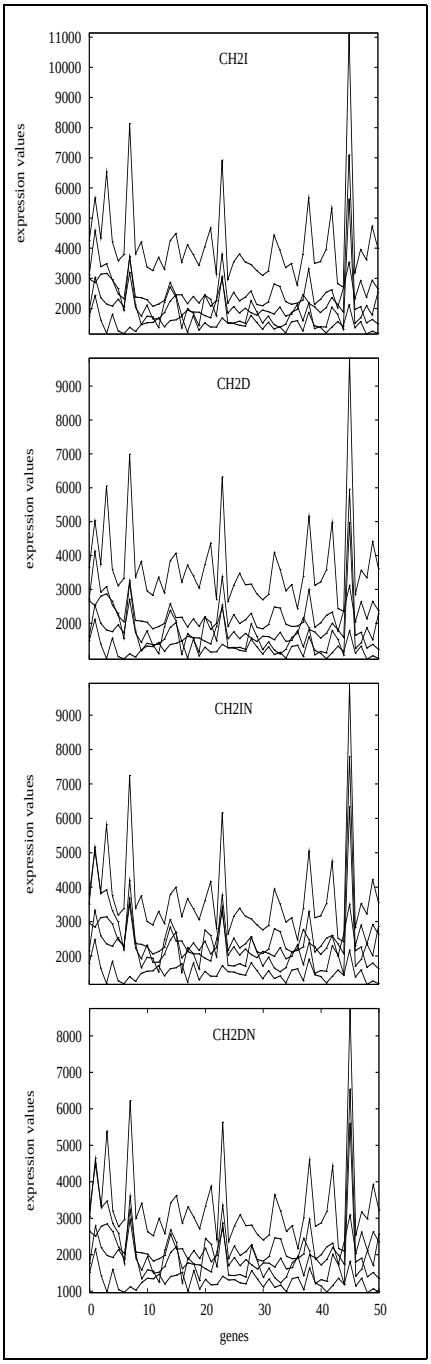


Figure 9: Time-Curves

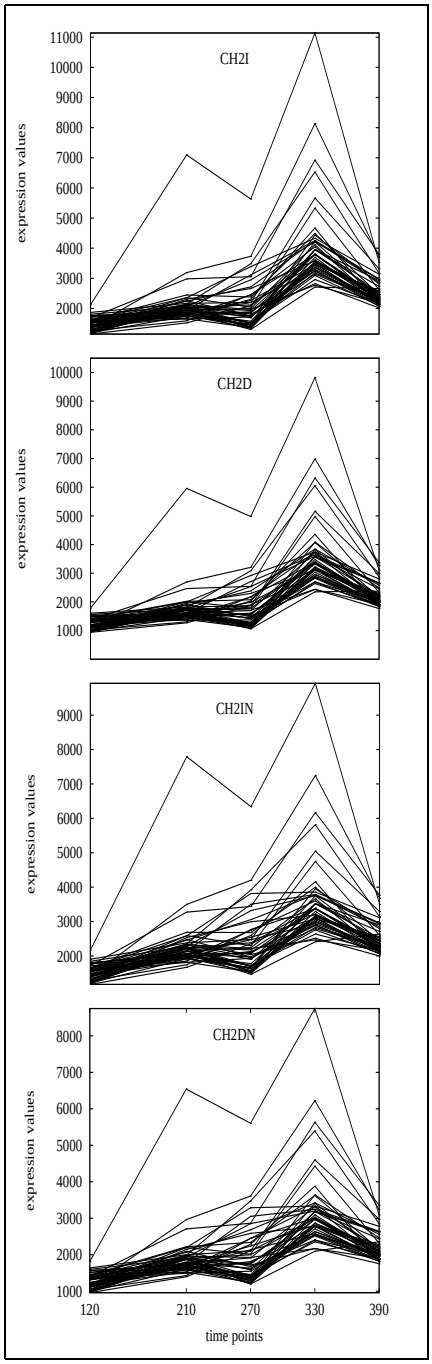


Figure 10: Gene-Curves

combination of the gene-sample-time dimensions.

The gene ontology (GO) project (www.geneontology.org) aims at developing three structured, controlled vocabularies (ontologies) that describe gene products in terms of their associated biological processes, cellular components and molecular functions in a species-independent manner. We used the yeast genome gene ontology term finder (www.yeastgenome.org) to verify the biological significance of triCluster's result. We obtained a hierarchy of GO terms for each gene within each cluster, for each of the three categories: processes, cellular components and gene functions. Table 2 shows the significant shared GO terms (or parents of GO terms) used to describe the set of genes in each cluster. The table shows the number of genes in each cluster and the significant GO terms for the process, function and component ontologies. Only the most significant common terms are shown. For example for cluster C_0, we find that the genes are mainly involved in ubiquitin cycle. The tuple $(n = 3, p = 0.00346)$ means that out of the 51 genes, 3 belong to this process, and the statistical significance is given by the p-value of 0.00346. Within each category, the terms are given in descending order of significance (i.e., increasing p-values). Further, only p-values lower than 0.01 are shown; the other genes in the cluster share other terms, but at a lower significance. From

Cluster	#Genes	Process	Function	Cellular Component
C_0	51	ubiquitin cycle (n=3, p=0.00346), protein polyubiquitination (n=2, p=0.00796), carbohydrate biosynthesis (n=3, p=0.00946)		
C_1	52	G1/S transition of mitotic cell cycle (n=3, p=0.00468), mRNA polyadenylylation (n=2, p=0.00826)	protein phosphatase regulator activity (n=2,p=0.00397) , phosphatase regulator activity (n=2, p=0.00397)	
C_2	57	lipid transport (n=2, p=0.0089)	oxidoreductase activity (n=7, p=0.00239), lipid transporter activity (n=2, p=0.00627), antioxidant activity (n=2, p=0.00797)	cytoplasm (n=41, p=0.00052), microsome (n=2, p=0.00627), vesicular fraction (n=2, 0.00627), microbody (n=3, p=0.00929), peroxisome (n=3, p=0.00929)
C_3	97	physiological process (n=76, p=0.0017), organelle organization and biogenesis (n=15, p=0.00173), localization (n=21, p=0.00537)	MAP kinase activity (n=2, p=0.00209), deaminase activity (n=2, p=0.00804), hydrolase activity, acting on carbon-nitrogen, but not peptide, bonds (n=4, p=0.00918), receptor signaling protein serine/threonine kinase activity (n=2, p=0.00964)	membrane (n=29, p=9.36e-06), cell (n=86, p=0.0003), endoplasmic reticulum (n=13, p=0.00112), vacuolar membrane (n=6, p=0.0015), cytoplasm (n=63, p=0.00169) intracellular (n=79, p=0.00209), endoplasmic reticulum membrane (n=6, p=0.00289), integral to endoplasmic reticulum membrane (n=3, p=0.00328), nuclear envelope-endoplasmic reticulum network (n=6, p=0.00488)
C_4	66	pantothenate biosynthesis (n=2, p=0.00246), pantothenate metabolism (n=2, p=0.00245), transport (n=16, p=0.00332), localization (n=16, p=0.00453)	ubiquitin conjugating enzyme activity (n=2, p=0.00833), lipid transporter activity (n=2, p=0.00833)	Golgi vesicle (n=2, p=0.00729)

Table 2: Significant Shared GO Terms (Process, Function, Component) for Genes in Different Clusters

the table it is clear that the clusters are distinct along each category. For example, the most significant process for C_0 is ubiquitin cycle, for C_1 it is G1/S transition of mitotic cell cycle, for C_2 it is lipid transport, for C_3 it is physiological process/organelle organization and biogenesis, and for C_4 it is pantothenate biosynthesis. Looking at function we find the most significant terms to be protein phosphatase regulator activity for C_1, oxidoreductase activity for C_2, MAP kinase activity for C_3, and ubiquitin conjugating enzyme activity for C_4. Finally, the clusters also differ in terms of cellular component: C_2 genes belong to cytoplasm, C_3 genes to membrane, and C_4 to Golgi vesicle.

These results indicate that triCluster can find potentially biologically significant clusters either in genes or samples or times, or any combinations of these three dimensions. Since the method can mine coherent subspace clusters in any 3D dataset, triCluster will also prove to be useful in mining temporal and/or spatial dimensions. For example, if one of the dimensions represents genes, another the spatial region of interest, and the third dimension the time, then triCluster can find interesting expression patterns in different regions at different times.

6. CONCLUSIONS

In this paper we introduced a novel deterministic triclustering algorithm called triCluster, which can mine arbitrarily positioned and overlapping clusters. Depending on different parameter values, triCluster can mine different types of clusters, including those with constant or similar values along each dimension, as well as scaling and shifting expression patterns. triCluster first constructs a range multigraph, which is a compact representation of all similar value ranges in the dataset between any two sample columns. It then searches for constrained maximal cliques in this multigraph to yield the set of biclusters for this time slice. Then triCluster constructs another bicluster graph using the biclusters (as vertices) from each time slice. The clique mining of the bicluster graph will give the final set of triclusters. Optionally, triCluster merges/deletes some clusters having large overlaps. We present a useful set of metrics to evaluate the clustering quality, and we evaluate the sensitivity of triCluster to different parameters. We also show that it can find meaningful clusters in real data. Since cluster enumeration is still the most expensive step, in the future we plan to develop new techniques for pruning the search space.

Acknowledgment

This work was supported in part by NSF CAREER Award IIS-0092978, DOE Career Award DE-FG02-02ER25538, and NSF grants EIA-0103708 and EMT-0432098.

7. REFERENCES

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

[2] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD Conference*, 2000.

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

[4] Z. Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.

[5] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *6th Annual Int'l Conference on Computational Biology*, 2002.

[6] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. In *3rd Annual Int'l Conference on Computational Biology, RECOMB*, 1999.

[7] Y. Cheng and G. M. Church. Biclustering of expression data. In *8th Int'l Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.

[8] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Science, USA*, 95(25):14863–14868, 1998.

[9] S. Erdal, O. Ozturk, D. Armbruster, H. Ferhatosmanoglu, and W. Ray. A time series analysis of microarray data. In *4th IEEE Int'l Symposium on Bioinformatics and Bioengineering*, May 2004.

[10] J. Feng, P. E. Barbano, and B. Mishra. Time-frequency feature detection for timecourse microarray data. In *2004 ACM Symposium on Applied Computing*, 2004.

[11] V. Filkov, S. Skiena, and J. Zhi. Analysis techniques for microarray time-series data. In *5th Annual Int'l Conference on Computational Biology*, 2001.

[12] J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society Series B*, 66(4):815, 2004.

[13] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, and R. Shamir. An algorithm for clustering cdnas for gene expression analysis. In *3rd Annual Int'l Conference on Computational Biology*, 1999.

[14] D. Jiang, J. Pei, M. Ramanathany, C. Tang, and A. Zhang. Mining coherent gene clusters from gene-sample-time microarray data. In *10th ACM SIGKDD Conference*, 2004.

[15] J. Liu and W. Wang. OP-cluster: clustering by tendency in high dimensional spaces. In *3rd IEEE Int'l Conference on Data Mining*, pages 187–194, 2003.

[16] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[17] C. S. Moller, F. Klawonn, K. Cho, H. Yin, and O. W. uer. Clustering of unevenly sampled gene expression time-series data. *Fuzzy Sets and Systems*, 2004.

[18] T. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pacific Symposium on Biocomputing*, 2003.

[19] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. Murali. A monte carlo algorithm for fast projective clustering. In *ACM SIGMOD Conference*, 2002.

[20] M. F. Ramoni, P. Sebastiani, and I. S. Kohane. Cluster analysis of gene expression dynamics. *Proceedings of the National Academy of Sciences, USA*, 99(14):9121–9126, July 2002.

[21] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In *Int'l Conference on Intelligent Systems for Molecular Biology*, 2000.

[22] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, Dec. 1998.

[23] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proceeedings of the National Academy of Science, USA*, 96(6):2907–2912, 1999.

[24] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Suppl.1):S136–S144, 2002.

[25] C. Tang, L. Zhang, A. Zhang, and M. Ramanathan. Interrelated two-way clustering: An unsupervised approach for gene expression data analysis. In *2nd IEEE Int'l Symposium on Bioinformatics and Bioengineering*, 2001.

[26] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *ACM SIGMOD Conference*, 2002.

[27] E. P. Xing and R. M. Karp. Cliff: clustering high-dim microarray data via iterative feature filtering using normalized cuts. *Bioinformatics*, 17(Suppl.1):S306–S315, 2001.

[28] 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.

[29] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: Capturing subspace correlation in a large data set. In *18th Int'l Conference on Data Engineering, ICDE*, 2002.

[30] K. Yeung and W. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.