Clustering Gene Expression Patterns

Amir Ben-Dor* Ron Shamir† Zohar Yakhini‡

July 14 1999

Abstract

Recent advances in biotechnology allow researchers to measure expression levels for thousands of genes simultaneously, across different conditions and over time. Analysis of data produced by such experiments offers potential insight into gene function and regulatory mechanisms. A key step in the analysis of gene expression data is the detection of groups of genes that manifest similar expression patterns. The corresponding algorithmic problem is to cluster multi-condition gene expression patterns.

In this paper we describe a novel clustering algorithm that was developed for analysis of gene expression data. We define an appropriate stochastic error model on the input, and prove that under the conditions of the model, the algorithm recovers the cluster structure with high probability. The running time of the algorithm on an n-gene dataset is $O(n^2(\log(n))^c)$. We also present a practical heuristic based on the same algorithmic ideas. The heuristic was implemented and its performance is demonstrated on simulated data and on real gene expression data, with very promising results.

1 Introduction

In any living cell that undergoes a biological process, different subsets of its genes are expressed in different stages of the process. The particular genes expressed at a given stage and their relative abundance are crucial to the cell's proper function. Measuring gene expression levels in different developmental stages, different body tissues, different clinical conditions and different organisms is instrumental in understanding biological processes. Such information can help the characterization of gene function, the determination of effects of experimental treatments, and the understanding of many other molecular biological processes.

Current approaches to measuring gene expression profiles include SAGE [34], RT/PCR [31], and hybridization based assays. In the latter, a set of oligonucleotides, or a set of appropriate cDNA molecules,

*Department of Computer Science & Engineering, University of Washington (amirbd@cs.washington.edu). Supported by the Program in Mathematics and Molecular Biology.

†Department of Computer Science, Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv 69978 ISRAEL. (shamir@math.tau.ac.il) http://www.math.tau.ac.il/∼shamir.

‡HPLI, Hewlett Packard Laboratories (zohary@hpl.hp.com).

is immobilized on a surface to form the hybridization array. When a labeled target DNA (or RNA) mixture is introduced to the array, target sequences hybridize to complementary immobilized molecules. The resulting hybridization pattern (detected, for example, by fluorescence) is indicative of the mixture's content. Hybridization arrays are thus used as molecular recognition tools for nucleic acids (see [8, 20, 23, 28, 26, 4, 24, 12].)

The development of the above mentioned methods over the last few years has led to tremendous acceleration in the rate at which gene expression pattern information is accumulated, cf. [21, 25, 10, 35, 19]. One can rapidly record the expression levels of thousands of genes under various conditions. Elucidating the patterns hidden in the data is a tremendous opportunity and challenge for functional genomics and proteomics. A first step in answering this challenge is via clustering techniques, which aim to identify sets of genes that "behave similarly" across the conditions.

In cluster analysis, one wishes to partition entities into groups based on given features of each entity, so that the groups are homogeneous and well-separated. Each group is called a *cluster*, and the partition is called *clustering*. Clustering problems arise in numerous disciplines including biology, medicine, psychology, economics and others. There is a very rich literature on cluster analysis going back over three decades (cf. [11, 17, 14, 27].) Numerous approaches were proposed to defining quality criteria for solutions, stipulating the type of clustering sought, and interpreting the solutions. Algorithmic approaches also abound. Most formulations of the problem are NP-hard, so the algorithmics emphasizes heuristics and approximation. For a good recent survey on clustering with an emphasis on graph theoretic and operations research approaches see [16].

Analyzing multi-conditional gene expression patterns with clustering algorithms involves the following steps:

- Determination of the gene expression data. The data can be represented by a real-valued *expression matrix* I where I_{ij} is the measured expression level of gene i in experiment (condition) j. Expression levels should ideally be absolute, but often only relative levels are available. The i-th row of the matrix is a vector forming the *expression pattern* of gene i.

- Calculation of a similarity matrix S. In this matrix the entry S_{ij} represents the similarity of the expression patterns for genes i and j. Many possible similarity measures can be used here. A good choice of measure depends on the nature of the biological question and on the technology that was used to obtain the data.

- Clustering the genes based on the similarity data or on the expression data. Genes that belong to the same cluster should have similar expression patterns, while different clusters should have distinct, well-separated patterns. The clustering algorithm is the main topic of this paper.

- Representations of the constructed solution. As hundreds or thousands of genes are involved, visualization tools are crucial for organizing, understanding and exploiting the results. Some examples of such tools will be given in Section 4.

Several algorithmic techniques were previously used in clustering gene expression data. Hierarchical clustering techniques, similar to those used in forming evolutionary trees, were shown to be valuable in [31, 35, 13, 32]. Self-Organizing Maps were used by Tamayo et. al [33]. Alon et. al [2] used deterministic annealing to perform hierarchical clustering. The latter study showed that the clustering can be applied to separate normal and tumor tissues and to differentiate tumor types based on gene expression pattern in each tissue. Graph theoretic approaches were also used for the related problem of clustering cDNAs based on their oligo-nucleotide fingerprints, [23, 18].

We shall describe in this paper a novel algorithm for the problem of clustering gene expression patterns. Unlike the hierarchical approaches mentioned above, our algorithm does not build a tree of clusters: Clusters are built and portrayed as unrelated entities. In contrast with self-organizing maps, it does not assume a given number of clusters and an initial spatial structure of them, but determines cluster number and structure based on the data. The most favorable algorithm may depend on the particular expression problem at hand, but there are certainly situations where neither a tree structure nor a particular number of clusters are sought. It is too early to determine which algorithm (if any) is better in more prevalent situations.

One advantageous property of our algorithm is that it can be analyzed probabilistically: We describe below a stochastic model for cluster formation and for data errors. Based on this model, we show that the algorithm reconstructs the clustering with high probability. The running time of the algorithm on an n-gene dataset is $O(n^2(\log(n))^c)$. We also present a practical heuristic based on the same algorithmic ideas, which was designed to reduce the running time and improve performance. The performance of that algorithm is demonstrated on simulated data and on real gene expression data, with very promising results.

We note that the algorithms below are not limited to gene expression applications and can be used in any biological or non-biological clustering problems.

The paper is organized as follows: In Section 2 we describe the stochastic model and the algorithm, analyze its complexity and prove its asymptotic properties. In Section 3 we present the heuristic algorithm and demonstrate its performance on simulated gene expression data. In Section 4 we apply it to actual gene expression data, and analyze its output.

A preliminary version of this paper was published in the Proceedings of the Third International Conference on Computational Molecular Biology (RECOMB '99) [3].

2 The Clustering Algorithm

In this section we describe the theoretical basis of our approach. For simplicity, we assume in this section that the similarity matrix, S, is a zero-one matrix, where a value of '1' corresponds to two genes that have a similar expression pattern, and '0' corresponds to non-similar patterns.

We approach the clustering problem at hand by studying a stochastic model of the data. The model assumes that the input data is obtained from the underlying cluster structure by "contamination" with random errors. We first describe the model assumptions, and then describe an appropriate clustering algorithm. Finally, we prove that with high probability, the algorithm succeeds in the clustering.

2.1 The Probabilistic Model

We assume that there exists a true biological partition of the genes into disjoint clusters, based on the functionality of the genes. Moreover, each cluster has a characteristic expression profile, and the genes within it have expression patterns which are variations of the cluster profile. These expression patterns are approximately measured using DNA arrays or other technologies. From the measured expression data, the similarity matrix S is computed. This complex process inevitably introduces errors into S (e.g., inter-cluster genes that show low similarity). We model these errors by assuming that S is a result of random contamination of the true clustering relation. To be specific, we assume that for any two genes the probability of computing the correct answer (whether the two genes are in the same cluster or not) is $1-\alpha$ (where $\alpha < \frac{1}{2}$ is some constant). With probability α we get the wrong answer.

We represent the input data using an undirected graph, G, where each gene is represented by a node, and edges connect genes that have similar expression pattern. A graph is called a *clique graph* if it is a disjoint union of complete graphs. A clique graph represents a possible clustering of the vertices (genes) - Each clique corresponds to a cluster that contains the corresponding genes. We denote by H the clique graph that represents the true clustering of the genes (underlying the input graph G). Given two graphs over the same set of vertices, $G(V,E)$ and $G'(V,E')$, we define the *distance* between them by $\Delta(G,G') = |E \triangle E'|$, where $E\Delta E' = \{E \setminus E'\} \cup \{E' \setminus E\}$.

Using the graph representation, our model assumes that the input graph was generated from H by flipping each edge/non-edge with probability α. Therefore, it makes sense to prefer clutering solutions (clique graphs) that imply fewer flips (errors). We present an efficient algorithm that with high probability returns a solution which is at least as good as the true clustering. That is, with high probability the output clique graph is no further from the input graph than H is.

The corrupted clique-graph model described above is similar to several probabilistic graph models. Kucera [22] considers a variety of graph partitioning problems, including bisection and coloring, on random graphs. He suggests algorithms for solving these with high probability and studies their expected complexity assuming some specific distributions on the input. Alon et. al [1] give an algorithm for finding a single large hidden clique in a random graph. Condon and Karp [6] consider the graph l-partition problem: partition the nodes of an undirected graph into l subsets of predefined sizes so that the total number of inter subset edges is minimal. They present an algorithm whose complexity is linear in the number of edges that solves the problem with high probability, on the planted l-partition model. Note that our problem is different: no predefined structure is given (any clique structure is, apriori, a possible candidate) and minimality with respect to inter cluster edges as well as intra cluster non-edges is sought.

We now give a formal definition of the model. We also refer the raeder to Figure 3 (A - C), for a visual impression of the model.

Definition 2.1:

(i) A *cluster structure* is a vector $S = (s_1, s_2, ..., s_d)$, where $s_i > 0$ and $\sum s_i = 1$. For a cluster structure S, let $\gamma(S)$ be the smallest entry and denote by $d(S)$ the dimension, d.

(ii) We say that an n-vertex clique graph has structure $S = (s_1, s_2, ..., s_d)$ if it consists of d disjoint cliques of sizes $s_1 n, s_2 n, \ldots, s_d n$.

∎

Definition 2.2: The random graph model $\mathcal{Q}(n, \alpha, S)$ (representing random corruption of clique graphs) is defined as follows: Given a clique graph H over n vertices with structure S, and a value $0 \leq \alpha < \frac{1}{2}$, the random graph $G_{H,\alpha}$ is obtained from H by randomly (1) removing each edge in H with independent probability α; (2) adding each edge not in H with independent probability α. ∎

Definition 2.3: Consider an algorithm $\mathcal{A}$ that takes an arbitrary graph G as input and returns a clique graph $\mathcal{A}(G)$ on the same vertex set. Let $\delta > 0$. We say that $\mathcal{A}$ *clusters* $\mathcal{Q}(n, \alpha, S)$ *with probability* $1 - \delta$ if when applied to the random graph $G_{H,\alpha}$, the output graph is, asymptotically, as good a solution as the original clique graph is, with probability $1 - \delta$. More precisely, we require that for n large enough, and for any clique graph H with structure S, we have

$$\mathbb{P}[\Delta(\mathcal{A}(G_{H,\alpha}), G_{H,\alpha}) \leq \Delta(H, G_{H,\alpha})] > 1 - \delta.$$

Here and throughout this section $\mathbb{P}$ denotes the relevant probability measure, which is clear from the context.

∎

2.2 The Algorithm

We introduce some notation, before presenting the algorithm. We use $D(p\|a)$ to denote the *relative entropy distance from* $(p, 1-p)$ *to* $(a, 1-a)$, that is, $D(p\|a) = p\log_2(p/a) + (1-p)\log_2((1-p)/(1-a))$. We use $k(\alpha)$ to denote $\lceil 2/D(1/2\|\alpha) \rceil$. All logs are natural unless explicit notation is used.

For a subset of vertices $U \subset V$ and $v \notin U$, let $\deg(v, U) = |\{(v, u) \in E : u \in U\}|$. Let $\langle W_1, \ldots, W_\ell \rangle$ be a partition of $U \subset V$. A vertex v outside U is *attracted* to W_i if W_i has the highest proportion of neighbors of v. That is, $\deg(v, W_i)/|W_i| \geq \deg(v, W_j)/|W_j|$ for $j = 1, \ldots, l$. The algorithm is described in Figure 1.

To analyze the algorithm we need the following theorem, due to Chernoff ([5], [9, Section 2.2]).

Theorem 2.4: *(Chernoff, 1952)*
Let $X \sim$ Binomial(N, p). Let $a < p < b$. Then $\mathbb{P}(X \geq bN) < \exp(-ND(b\|p))$, and $\mathbb{P}(X \leq aN) <$ $\exp(-ND(a\|p))$.

We also need a very crude sampling lemma:

Parallel Classification with Cores (PCC)

Input:

A graph $G(V, E)$ over n vertices.
A constant $\alpha < \frac{1}{2}$ /* an upper bound on the contamination error */
A constant $\delta > 0$ /* the maximal tolerated failure probability of the algorithm */
A constant $\gamma > 0$ /* a lower bound on $\gamma(S)$ */
Let $m = \lceil \frac{1}{\gamma} \rceil$.

Algorithm:

1. Uniformly draw a subset U_1 of $2m \cdot k(\alpha) \log\log(n)$ vertices from V.
2. Uniformly draw a subset U_2 of $2m \cdot k(\alpha) \log(n)$ vertices from $V \setminus U_1$.
3. For every partition $W = \langle W_1, \ldots, W_\ell \rangle$ of U_1 into at most m subsets do:
   (a) Cluster the vertices of $U_1 \cup U_2$, based on the partition W - The i-th cluster, X_i, will contain the vertices in W_i and all the vertices from U_2 that are attracted to W_i. Denote the resulting partition by $X(W) = \langle X_1, \ldots X_\ell \rangle$.
   (b) In a similar manner cluster all of the n vertices based on the partition X of $U_1 \cup U_2$. The i-th cluster, Y_i, will contain the vertices in X_i and all vertices from $V \setminus (U_1 \cup U_2)$ that are attracted to X_i. Denote the resulting partition by $Y(W) = \langle Y_1, \ldots Y_\ell \rangle$. This partition determines a clique graph over V.
4. Amongst all these clique graphs, choose the one which is closest (in the symmetric difference sense) to the input graph.

Figure 1: PCC algorithm

Lemma 2.5: *Consider n objects of d different colors, where each color is represented by at least n/m objects. If s objects are sampled uniformly and independently without replacement, then*

$$\mathbb{P}\left(\begin{array}{c} \textit{The sample contains} \geq s/2m \\ \textit{representatives of each color} \end{array} \right) > 1 - \delta,$$

provided that $16m^2 \log(d/\delta) \leq s \leq \frac{n}{4m}$.

Proof: Call a sample as above *bad* if it does not satisfy the condition for a fixed color A.

$$p = \mathbb{P}(\text{ bad sample }) \leq \mathbb{P}(X < s/2m)\,,$$

where $X \sim \text{Binomial}\left(s, \frac{(n/m)-s}{n}\right)$. This is true since even with no replacement the proportion of A-colored elements left in the pile in each trial is more than $\frac{(n/m)-s}{n}$. Therefore, by the Chernoff bound above, and assuming $n > 4ms$,

$$\begin{aligned} p &< \exp\left(-s \cdot D(\frac{1}{2m} \| \frac{3}{4m})\right) \qquad (1) \\ &\leq \exp\left(-\frac{s}{16\log(2)m^2}\right). \end{aligned}$$

The inequality in (2) follows from the general inequality (cf. [7]): $D(p\|q) \geq (1/\ln(2)) \cdot (p-q)^2$. This last expression is less than δ/d by our assumption on the sample size s. A union over all colors yields the stated result. ∎

Theorem 2.6: *Let S be a cluster structure and let $\alpha < 1/2$. For any fixed $\delta > 0$ the above algorithm clusters $\mathcal{Q}(n, \alpha, S)$ with probability $1-\delta$. The time complexity of the algorithm is $O\left(n^2 \cdot \log(n)^c\right)$, where c is a constant that depends on α and on $\gamma(S)$.*

Proof: Since $m = \lceil \frac{1}{\gamma(S)} \rceil$, $d(S) \leq m$. m is considered a constant for our setup. Let $T = \langle T_1, \ldots T_m \rangle$ be the partition of V that represents the underlying clusters, where some clusters may be empty. For a vertex $v \in V$ let $i(T,v)$ be defined by $v \in T_{i(T,v)}$. Let $\eta > 0$ (η will be related to the tolerated failure probability, δ, at the end). Recall that $k(\alpha) = \lceil 2/D(1/2\|\alpha) \rceil$. In our analysis below we refer to the steps of Figure 1.

1. Uniformly draw a subset U_1 of vertices of size $2m \cdot k(\alpha) \log\log(n)$. If n is large enough, namely: $\log\log(n) > 8mk(\alpha)\log(1/\eta)$ and $n > 8m^2 k(\alpha) \log\log(n)$ we know (by Lemma 2.5) that with probability $1-\eta$ each color has at least $k(\alpha)\log\log(n)$ representatives in this chosen subset.

2. Uniformly over the subsets of $V \setminus U_1$ draw a subset U_2 of vertices with $2m \cdot k(\alpha)\log(n)$ elements. Again, for n large enough, with probability $1-\eta$ each color has at least $k(\alpha)\log(n)$ representatives in this subset.

3. Consider all partitions of U_1 into m subsets (for n large enough there are less than $\log(n)^{2m\log(m)\cdot k(\alpha)}$ of them). Denote each such partition by $W = \langle W_1, \ldots, W_m \rangle$ (some subsets may be empty). Run the following enumerated steps starting with all these partitions. For the analysis focus on a partition where each W_i is a subset of a distinct true cluster T_j. Such a partition is, indeed, considered, since we are considering all partitions. For this case we can further assume, without loss of generality, that for each i we have $W_i \subset T_i$.

   (a) Start with sets $X_i = W_i$. For all $u \in U_2$ let $i(X,u)$ be the index that attains the maximum ($1 \leq i \leq m$) of $\deg(u, W_i)/|W_i|$. Add u to that set. Let $W(u) = W_{i(T,u)}$. The collection of edges from u to $W(u)$ are independent Bernoulli$(1-\alpha)$ (the drawings of U_1 and U_2 were independent of everything else). Therefore $\deg(u, W(u)) \sim$ Binomial$(|W(u)|, 1-\alpha)$. Using the Chernoff bound stated above we therefore have

$$\begin{aligned} \boldsymbol{P}\left(\deg(u, W(u)) \leq \frac{|W(u)|}{2}\right) &< \exp(-|W(u)|D(\tfrac{1}{2}\|\alpha)) \\ &< \log(n)^{-k(\alpha)D(\frac{1}{2}\|\alpha)} \qquad (2) \\ &< \log(n)^{-2}, \qquad (3) \end{aligned}$$

   where $|W(u)| \geq k(\alpha)\log\log(n)$ justifies (2). Similarly, for $i \neq i(T,u)$, we have

$$\deg(u, W_i) \sim \text{Binomial}(|W_i|, \alpha),$$

and thus

$$\begin{aligned} \boldsymbol{P}(\deg(u, W_i) \geq |W_i|/2) &< \exp(-|W_i| D(\tfrac{1}{2} \| \alpha)) \\ &< \log(n)^{-2}, \end{aligned} \tag{4}$$

whence $i(X, u) = i(T, u)$ with high probability: $\boldsymbol{P}(i(X, u) \neq i(T, u)) < m \log(n)^{-2}$. Finally, by a union bound

$$\boldsymbol{P}(i(X, u) \neq i(T, u) \text{ for some } u \in U_2\ ) < 2m^2 \cdot k(\alpha) \log(n)^{-1}. \tag{5}$$

(b) Focusing on the part of the measure space where no error was committed in the previous steps (in particular, all vertices were assigned to their original color), we now have m subsets of vertices $X_i \subset T_i$, $i = 1...m$, each of size at least $k(\alpha) \log(n)$, unless the corresponding T_i is empty. We take all other vertices and classify them using these subsets, as in the previous step. Let the resulting partition be $Y == \langle Y_1, \ldots, Y_m \rangle$ and for vertices $v \in V$ let $i(Y, v)$ be defined by $v \in Y_{i(Y,v)}$. Observe that all edges used in this classification are independent of the algebra generated by everything previously done. This is true since in the previous step only edges from U_2 to U_1 were considered, and these are of no interest here. Therefore, the equivalents of (3) and (4) hold, yielding

$$\boldsymbol{P}(i(Y, v) \neq i(T, v) \text{ for any } v \in V\ ) < 2m^2 \cdot k(\alpha) n^{-1}. \tag{6}$$

4. Amongst all outputs of the above, choose the partition which is closest (in the symmetric difference sense) to the input graph.

The total probability of failure in this process is estimated as follows

$$\begin{aligned} &\boldsymbol{P}\left(\begin{array}{c} \text{The original partition } \ V = \bigcup_{i=1}^m T_i \\ \text{is not one of the outputs} \end{array} \right) \\ &\quad \leq \ 2\eta + 2m^2 \cdot k(\alpha) \left(n^{-1} + \log(n)^{-1} \right), \end{aligned} \tag{7}$$

which is arbitrarily small for large n and if η is chosen appropriately.

As noted above, we have less than $\log(n)^{2m \log(m) \cdot k(\alpha)}$ possible partitions of U_1. Each such partition leads to a clustering of all vertices in V, using the core clusters X_i , $i = 1...m$. For each partition $O(n \log(n))$ edges are considered in the classification step. Each edge is considered at most once, as sums of disjoint edge subsets are compared to a threshold. Computing the distance of each of the clique graphs produced to the input graph requires $O(n^2)$ operations. Thus the total time complexity of the algorithm is $O(n^2 \cdot \log(n)^{2m \log(m) \cdot k(\alpha)})$. ∎

3 A Practical Heuristic

3.1 Algorithm CAST

In this section we take a more practical approach, and present a novel and simple clustering heuristic, called `Cluster Affinity Search Technique`, or, in short, `CAST`. The algorithm uses the same ideas as the theoretical algorithm described in Theorem 2.6: it uses average similarity (affinity) between unassigned vertices and the current cluster core to make its next decision. However, it differs from the theoretical algorithm in several aspects: (1) The theoretical algorithm repeats the same process for many initial seeds. In order to avoid the repetition, we instead use "cleaning" steps to remove spurious elements from cluster cores. (2) `CAST` adds (and removes) elements from the current core one at a time (and not independently, as in the theoretical algorithm). Heuristically, this helps by strengthening the constructed core, thus improving the decision base for the next step. (3) `CAST` handles more general input: Similarities can be real-valued, and a user-specified *affinity threshold* parameter t determines what affinity level is considered significant. This parameter influences the number and sizes of the produced clusters.

The input to the algorithm is a pair $\langle S, t\rangle$, where S is a real symmetric n-by-n similarity matrix ($S(i,j) \in [0,1]$), and t is the affinity threshold. The clusters are constructed one at a time. The currently constructed cluster is denoted by C_{open}. We define the *affinity* of an element x with respect to the current cluster by $a(x) = \sum_{y \in C_{\text{open}}} S(x,y)$. We say that an element x has *high affinity* if $a(x) \geq t|C_{\text{open}}|$. Otherwise, x has *low affinity*. `CAST` alternates between adding high affinity elements to C_{open}, and removing low affinity elements from it. When this process stabilizes C_{open} is closed and a new cluster is started. A pseudo-code of the algorithm is given in Figure 2.

To save time, `CAST` avoids the repetition employed in the theoretical algorithm. Instead, each cluster is constructed incrementally, and addition of true cluster members will strengthen the cluster. This is done at the risk of forming an incorrect seed, which does not correspond to any true cluster. The removal of elements allows one to (heuristically) strengthen the quality of the constructed seed and root out of the cluster members that were incorrectly added at early stages. The strong dependence between the algorithm's decisionsin the different stages (same edges are considered more than once) makes its analysis very hard, and we will support it solely by empirical preformance results.

We remark that the "cleaning" steps in `CAST` serve to avoid a common shortcoming shared by many popular clustering techniques (such as `single-linkage`, `complete-linkage`, `group-average`, and `centroid`, see [14, ch. 4]): due to their "greedy" nature, once a decision to join two elements in one cluster is made, it cannot be undone.

An additional improvement to `CAST` is the following: For each $v \in V$, let $C(v)$ be the cluster it is assigned to by `CAST`. For a cluster C and a vertex v, define $a(v,C) = \frac{1}{|C|}\sum_{u \in C} S(u,v)$. The final clustering of `CAST` guarantees that for every vertex v, $a(v, C(v)) \geq t$, but not necessarily that

$$a(v, C(v)) \geq a(v, C) \quad \text{for every other cluster } C. \tag{8}$$

After `CAST` halts, a series of moving steps are performed. For each vertex v, if there is a cluster C in the

Clustering Affinity Search Technique

- Input: An n-by-n similarity matrix S, and an affinity threshold t.
- Initialization:

$\mathcal{C} \leftarrow \emptyset$ /* The collection of closed clusters */
$U \leftarrow \{1, \ldots, n\}$ /* Elements not yet assigned to any cluster */

- **while**$(U \neq \emptyset)$ **do**

  $C_{\text{open}} \leftarrow \emptyset$ /* Start a new cluster */
  $a(\cdot) \leftarrow 0$ /* Reset affinity */
  Repeat steps ADD and REMOVE as long as changes occur:

  ADD: **while** $\max\{a(u)|u \in U\} \geq t|C_{\text{open}}|$ **do** /* U contains a high affinity element*/
  Pick an element $u \in U$ with maximum affinity.
  $C_{\text{open}} \leftarrow C_{\text{open}} \cup \{u\}$ /* Insert u into C_{open} */
  $U \leftarrow U \setminus \{u\}$ /* Remove u from U */
  For all $x \in U \cup C_{\text{open}}$ set $a(x) = a(x) + S(x, u)$ /* Update the affinity */

  REMOVE: **while** $\min\{a(u)|u \in C_{\text{open}}\} < t|C_{\text{open}}|$ **do** /* C_{open} contains a low affinity element*/
  Pick an element $v \in C_{\text{open}}$ with minimum affinity.
  $C_{\text{open}} \leftarrow C_{\text{open}} \setminus \{v\}$ /* Remove v from C_{open} */
  $U \leftarrow U \cup \{v\}$ /* Insert v into U */
  For all $x \in U \cup C_{\text{open}}$ set $a(x) = a(x) - S(x, v)$ /* Update the affinity */

  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_{\text{open}}\}$ /* Close the cluster */

  end

- Done, return the collection of clusters, $\mathcal{C}$.

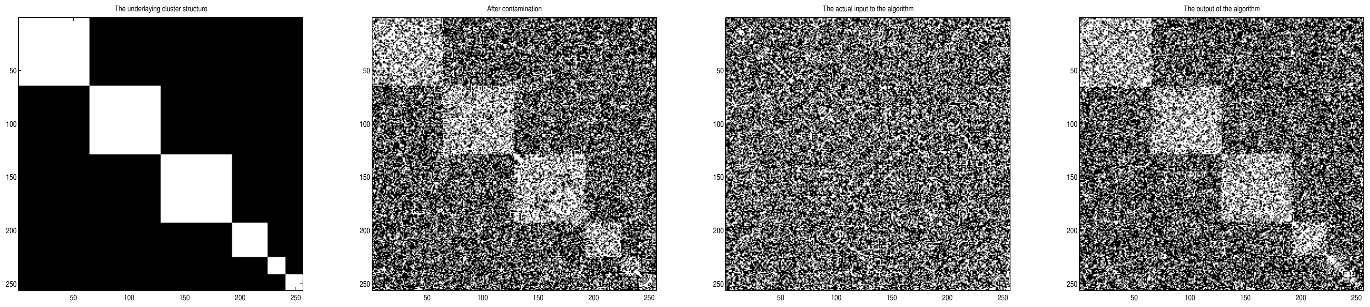
Figure 2: CAST Algorithm

---

current structure, such that $a(v, C) > a(v, C(v))$, then v is moved to C. This process is repeated until it converges or until some maximum of iterations is completed. If the process does indeed converge (8 holds for all vertices. In such a case we call the cluster structure *stable*.

3.2 Simulation Results

As is often the case with practical heuristics, it is very hard to prove rigorous performance bounds for `CAST`. Instead, we test its performance on randomly generated data. Recall (Definition 2.2) that the corrupted clique graph random graph model is specified by three parameters: a cluster structure S, an error probability α, and a size parameter n. For different choices of these parameters, we perform the following steps, mimicking the probabilistic model $\mathcal{Q}(n, \alpha, S)$: (A) Let H_0 be the clique graph with cluster structure S which has an adjacency matrix as in Figure 3(A) (the cluster blocks are along the diagonal). Permute the vertices of H_0 at random to obtain some graph H with cluster structure S. Flip each edge with independent probability α, obtaining the graph $G = G_{H,\alpha}$. (B) Apply `CAST` to the adjacency matrix



A B C D

Figure 3: A visual representation of of the simulation process. Cluster structure is $(1/4, 1/4, 1/4, 1/8, 1/16, 1/16)$, $n = 256$. A: the adjacency matrix of the original clique graph before the introduction of errors. Position (i, j) is black iff i and j belong to the same cluster. B: the same matrix after the introduction of errors. Note that the cluster structure is still visible for all but the smallest clusters. C: Same as B but entries are randomly permuted. This is actual input to the algorithm. The challenge is to reconstruct B - and hence the clusters - from C. D: Matrix C reordered according to the solution produced by the algorithm. With the exception of perhaps the smallest clusters, the essential cluster structure is reconstructed.

of G, using affinity threshold $t = 0.5$.

For each such trial we compute the closeness between the suggested cluster structure and the original clusters represented by the clique graph H. Two quantitative criteria for closeness were used: matching coefficient and Jaccard's coefficient [14, p. 41]). For completeness we define the above mentioned similarity coefficients. Given two binary matrices A and B of the same dimensions, let N_{ij} be the number of entries on which A and B have values i and j, respectively. The *matching coefficient* between the two matrices is simply the ratio of the total number of entries on which the two matrices agree, to the total number of entries: $(N_{00} + N_{11})/(N_{00} + N_{01} + N_{10} + N_{11})$. The Jaccard coefficient is the corresponding ratio when "negative" matches (N_{00}) are ignored: $N_{11}/(N_{01} + N_{10} + N_{11})$. Both criteria were applied to the adjacency matrices of the original cluster structure and the one suggested by the algorithm. Jaccard coefficient is more appropriate when the clusters are relatively small, as in that case N_{00} will be the dominant factor even if the solution is very far from the true one.

Visual inspection, such as examplified in Figure 3 can also be used to get a feeling for the algorithm's performance.

Table 1 reports results of the `CAST` algorithm on simulated data generated with various parameter values. For each set of parameters, in this and all other simulations results reported later, at least 100 random instances were solved. Note that even with very high noise ($\alpha = 0.4$), the quality of the solution is quite good.

Figure 4 presents results for simulations in a range of values on n. Since the data is not normally distributed, boxplots [29] are used to show the distribution. (Boxplots show a box spanning the 25th

Cluster structure (S)	n	α	Matching coeff.	Jaccard's coeff.
$\langle 0.4, 0.2, 0.1, 0.1, 0.1, 0.1\rangle$	500	0.2	1.0	1.0
$\langle 0.4, 0.2, 0.1, 0.1, 0.1, 0.1\rangle$	500	0.3	0.999	0.995
$\langle 0.4, 0.2, 0.1, 0.1, 0.1, 0.1\rangle$	500	0.4	0.939	0.775
$\langle 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1\rangle$	1000	0.3	1.0	1.0
$\langle 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1\rangle$	1000	0.35	0.994	0.943

Table 1: Performance of the `CAST` algorithm on simulated data, randomly generated according to the model in Definition 2.2.

percentile to the 75th percentile of the data. The middle line indicates the median. If the distance between the 75th percentile and the 25th percentile is denoted by I, then the horizontal bars mark the furthest data point on each side that is less than 1.5I from the end of the box. Any data point outside the bars is individually graphed.) As expected, the number of entries needed to obtain a certain solution quality increases sharply as α gets close to .5. In contrast, for the cluster structure used in the simulation, even with a relatively high $\alpha = 0.3$, several hundred entries suffice to obtain extremely accurate clustering.

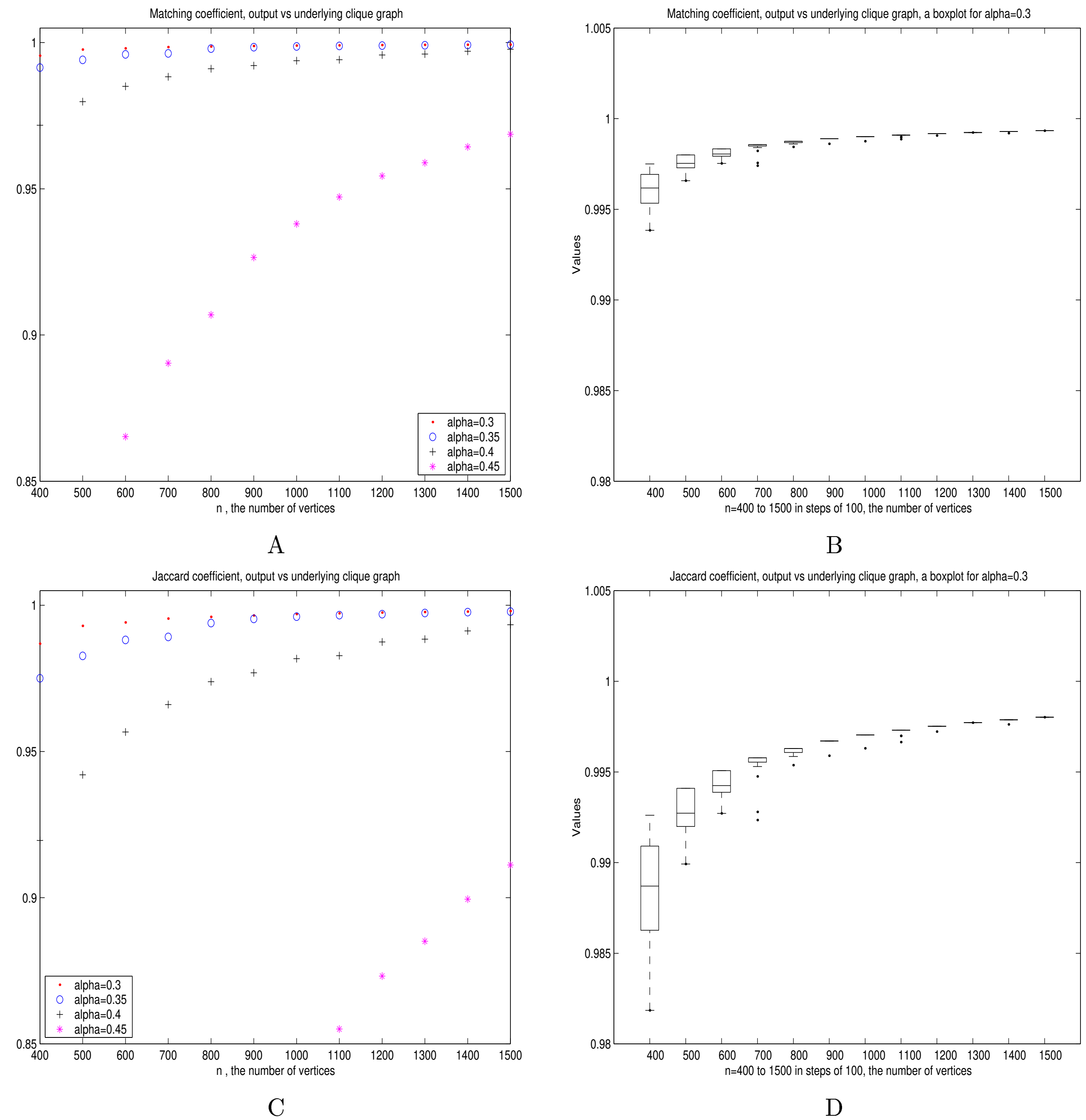


Figure 4: Simulation results for the cluster structure $S = [1/2, 1/4, 1/8, 1/16, 1/16]$. In all plots the x axis is the number of clustered entries. A and C: Matching and Jaccard coefficients mean values for several error parameters. B and D: Matching and Jaccard coefficients for $\alpha = 0.3$. Distribution is presented using boxplots.

3.3 Implementation

The software, including the `CAST` algorithm, the simulator of synthetic data and the visualization tools, was implemented using MATLAB. On an 1246×146 expression matrix (data from [21], to be discussed in Section 4.1 below) one clustering execution takes about ten seconds on a HP Vectra XU 6/180MHz with 96Mb RAM (not including a one time preprocessing step that computes the similarity matrix). Figure 5 presents running times for synthetic data with very high noise levels, on the same computer configuration. Note the sharp increase at about 1200 entries, where cache memory boundary is exceeded.

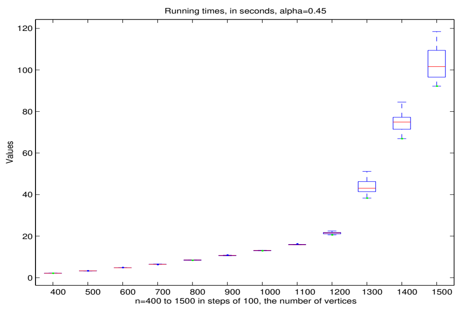


Figure 5: Running times for the `CAST` algorithm on simulated data. The cluster structure is $[1/2, 1/4, 1/8, 1/16, 1/16]$; $\alpha = 0.45$. x axis: problem size. y axis: running time in seconds. Distribution is shown using boxplots.

The similarity matrix of larger gene expression data can take large memory space. When data is too large to be handled in Matlab on a given machine configuration a procedure that computes similarity values whenever they are needed is employed. Since `CAST` accesses some entries of the matrix more than once, this increases the computation time. Expression patterns from a $\sim$ 6000 gene dataset (data from [10], results not reported here) was analyzed in this manner, requiring a running time of about two hours.

4 Applications to Biological Data

In this section we give some examples of the performance of the algorithm on real gene expression data sets. These examples demonstrate the usefulness of analysis tools in general and clustering tools, in particular, in enabling hypotheses generation and the definition of research directions based on gene expression data.

4.1 Temporal Gene Expression Patterns

We first analyze the data studied by Wen et. al. [35]. In this paper the authors study the relationship among expression patterns of genes involved in the rat Central Nervous System (CNS), measured during the development of the rat's CNS. Gene expression patterns for 112 genes were measured (using RT/PCR [31]) in cervical spinal cord tissue, at nine different developmental time points. This yields a 112×9 matrix of gene expression data. Expression levels per each gene were normalized by setting the maximum of every row to 1. Genes were categorized using prior knowledge into four families: Neuro-Glial Markers (NGMs), Neurotransmitter Receptors (NTRs), Peptide Signaling (PepS) and a fourth family containing all the rest, called Diverse (Div). Some families were further divided into subfamilies based on biological knowledge.

To capture the temporal nature of this data, the authors added for each gene the difference between the values at each two consecutive time points as an extra data point between them, thereby transforming each 9-dimensional expression vector into a 17-dimensional vector. This transformation enhances the similarity between genes with closely parallel, but offset, expression patterns. Euclidean distances between the augmented vectors were computed, yielding a 112×112 distance matrix. Next, A phylogenetic tree was constructed for this distance matrix (using FITCH [15]). Finally, cluster boundaries were determined by visual inspection of the resulting tree. A good correlation between the resulting clusters and the a-priori family information was observed.

We analyzed the same data in the following way. The raw expression data was first normalized and expression levels were augmented by the derivative values as in [35]. Then, a similarity matrix was computed using L_1 distance between the resulting 17-dimensional vectors. A hands-off version of our algorithm, which automatically searches for a good affinity threshold value, was applied to the similarity matrix (the eventual affinity threshold for the presented data was $t = 0.647$). Eight clusters were obtained. The results are depicted in Figure 6. Since the partition into clusters is assumed to be known, this experiment was done mainly for validation of the algorithm. Note that all clusters, perhaps with the exception of cluster 1, manifest clear and distinct expression patterns. Moreover, the agreement of most clusters with the prior biological clustering is quite good. (Note, for example, a single NGM in cluster #5 that is dominated by NTRs.) Our software enables visual comparison of the computed clustering solution to a user-defined partition into families. The user can also instruct the software to perform a subclustering of a single cluster. This was done here for cluster #1 and provided good separation both in terms of temporal patterns and in terms of agreements with the given gene subfamilies (results not shown).

4.2 Multi-Condition Expression Analysis

Clustering gene expression patterns is useful even when experiments' enumeration has no physical meaning (as opposed to temporal patterns). Kim et. al [21] studied gene regulation mechanisms in the nematode *C. elegans* using cDNA microarray hybridization assays. The data analyzed here consisted of expression levels of 1246 genes in 146 experiments. Some experiments are parts of time courses. Most experiments compare certain mutants to a reference cell (normal, at a fixed developmental stage). Expression levels were measured relative to the reference level, so each matrix entry value has the form $\log\left(\frac{\text{Red}}{\text{Green}}\right)$, representing the log-ratio of the intensities of the two dyes at the given array spot. Since the data is not temporal, we used Pearson correlation to evaluate the similarity of expression patterns here.

Figure 7 summarizes some of our results. The algorithm formed 40 clusters of size at least 5, the largest being of size 31. Clusters of four or fewer entries were ignored. Only very few genes out of the 1246 were classified into families by prior biological knowledge. These include genes of sperm proteins and yeast genes used as control. The algorithm clustered these families quite well into few homogeneous clusters

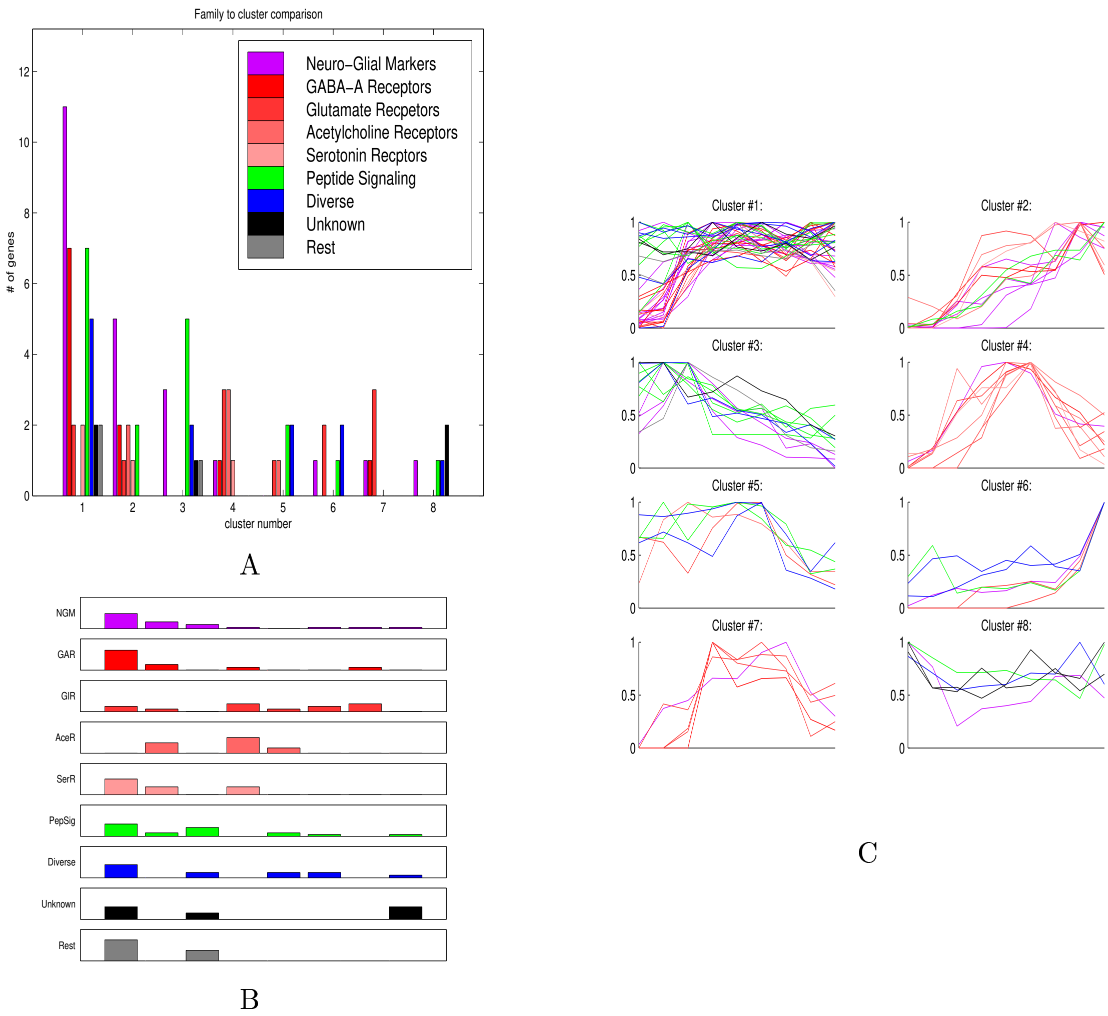


Figure 6: Results of the CAST algorithm on the temporal gene expression data of Wen et. al. The solution generated by the algorithm is compared to the prior classification. A: The composition of the eight suggested clusters, in terms of the biologically defined families. x axis: the cluster number. y axis: number of genes from each subfamily in each cluster, color coded according to the legend. NTRs are subdivided into four subfamilies as detailed in the legend, and are shown in different shades of red. B: The distribution of each subfamily among the eight clusters. Each box depicts the distribution of a different subfamily. x axis: cluster number. C: The expression patterns of genes in each of the clusters. The graphs are color coded as in A.

(see Figure 8). One example of the potential use of the algorithm's results is exemplified in Figure 7: A six-gene cluster containing two growth related genes and four anonymous genes was suggested. This raises the possibility that the other four genes are also growth related. This hypothesis is currently investigated experimentally.

Another direction of using the software is to single out conditions that have the largest effect on a

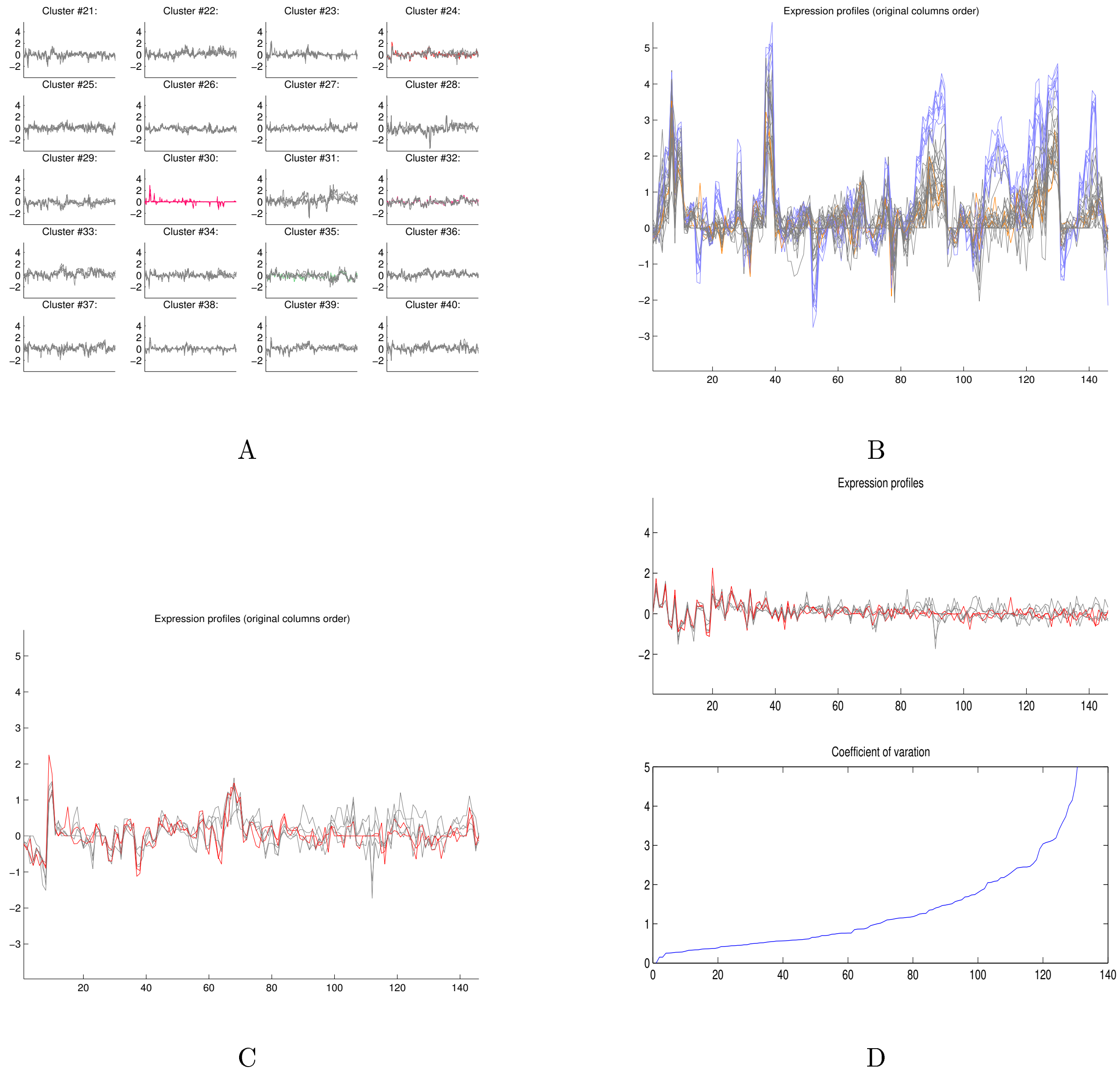


Figure 7: Some results of the `CAST` algorithm on the nematode gene expression data of Kim et. al. A: expression patterns for clusters #21 to #40. x axis: conditions (matrix columns) in arbitrary order. y axis: intensity level in $\log\left(\frac{\text{Red}}{\text{Green}}\right)$ scale. Most of the genes functions are unknown, so only few genes are color coded. Blue: sperm genes; red: yeast genes (control) ; gray: unknown. Note the homogeneity of cluster #30. B: expression patterns of the genes in cluster #1, consisting of 31 genes. C: Expression patterns of the six genes in cluster #24. This cluster contains two growth related genes, *lin-15* and *E2F*. This suggests the hypothesis that the other four members of this cluster have related function. D: Top: Same data as C, but conditions along the x axis are reordered in increasing values of their coefficient of variation. Bottom: The coefficient of variation for each condition in the corresponding order.

cluster. Figure 7 exemplifies this on one cluster: For each condition, the coefficient of variation of the values of the genes in the cluster is calculated. The purpose is to identify the conditions that characterize the common behavior of the elements of the cluster, for further inference and for biological interpretation. The software also allows the user to rebuild a cluster or clusters, taking into account only conditions for which the coefficient of variation was below a certain threshold.

For time courses it makes sense to use other similarity measures when the corresponding sub matrices are clustered. Clustering the columns (rather than the rows) of the expression matrix is also possible and contains biologically meaningful information.

The software also supports the generation of a report that contains the clustering results. Point and click operations for displaying cluster members and for interactively analyzing specific clusters are also supported. Cluster #1 (Figure 7) contained all known sperm proteins. The list of members are presented in Figure 8.

Some of the experiments performed by Kim et. al were time-course experiments. In one of these let23 deficient samples were measured in 7 time points during the 30-40 hours of the nematode's development process. Expression levels were determined by differential-labeling in comparison to a uniform reference sample. Two experiments were performed for each time point. The raw data consists of average values. The similarity measure used here was the L_1 distance between the vectors, augmented by the vector of differences as explained in Section 4.1, giving 13-dimensional vectors. In Figure 9 three of the clusters of genes from this set of experiments are depicted, showing strong homogeneity in expression patterns and in the known gene family identification.

```
Cluster number 1, 31 members
---------------------
142: YK411D12 *K07F5.1 major sperm protein                (sperm        )
149: YK411E07 *C09B9.6 major sperm protein                (sperm        )
177: YK411G11 *C09B9.6 major sperm protein                (sperm        )
181: YK411H03 *ZK1248.6 major sperm protein               (sperm        )
199: YK412A09 - major sperm protein                       (sperm        )
211: YK412B09 *ZK1248.6 major sperm protein               (sperm        )
239: YK412E01 *C09B9.6 major sperm protein                (sperm        )
281: YK412H07 *ZK1251.6 major sperm protein               (sperm        )
113: YK411B07 *F25F2.1 CELK06790, Protein kinase          (kinase       )
139: YK411D09 *C09B9.4 CELK02836, ser/thr protein kinase  (kinase       )
897: YK419C11 *T05A7.6 CELK00757, casein kinase           (kinase       )
105: YK411A11 *K01D12.7 CELK03798                         (Rest         )
116: YK411B10 - CELK06789                                 (Rest         )
129: YK411C11 - CELK05626                                 (Rest         )
153: YK411E11 - CELK03350, neural varian mena+++          (Rest         )
154: YK411E12 - neurogenin trans. fact.                   (Rest         )
159: YK411F05 *F53B6.4 CELK04342                          (Rest         )
179: YK411H01 - CELK06757                                 (Rest         )
195: YK412A05 - no 3-tag                                  (Rest         )
215: YK412C01 - CELK06801                                 (Rest         )
228: YK412D02 *C25G4.6 genpept:gi|1947007 (AF000197)      (Rest         )
257: YK412F07 - CELK05259                                 (Rest         )
262: YK412F12 - CELK02830                                 (Rest         )
264: YK412G02 *F36A2.10 >swiss:Q09167
      CL4_RAT INSULIN-INDUCED GROWTH RESPONSE PROTEIN CL-4 (Rest         )
876: YK419B02 *F49C12.15 CELK00579                        (Rest         )
922: YK419E12 - CELK03882                                 (Rest         )
948: YK419H02 - no 3-tag                                  (Rest         )
974: YK420B04 - CELK05963, Prot, Tyr. Phos.               (Rest         )
977: YK420B07 - Prot. Tyr. Phosph.                        (Rest         )
1169: YK422B07 - CELK01412, protein tyrosine phosphatase  (Rest         )
1204: YK422E06 - CELK00579                                (Rest         )
End of Cluster  number 1, 31 genes/experiments, total.
```

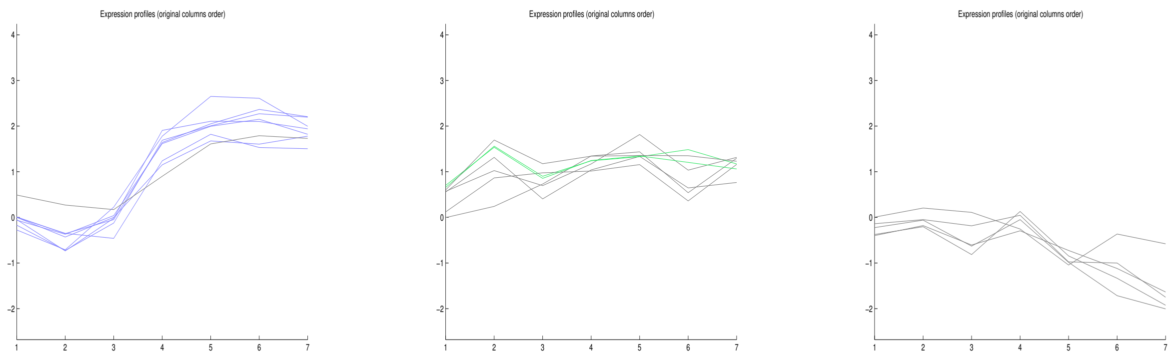
Figure 8: The list of members for cluster #1. This cluster contained all the known sperm proteins out of the 1246 genes. The software produces such reports per the user's request.

4.3 Tissue clustering

Another interesting application of clustering in gene expression was demonstrated recently by Alon et. al [2]. The authors describe an analysis of gene expression data obtained from 40 tumor and 22 normal colon samples. The study used absolute measurement of more than 6500 human genes, using Affymetrix oligonucleotide hybridization array technology. The authors describe a clustering algorithm, based on deterministic annealing [30]. This method is hierarchical and clusters are determined from an intermediate binary tree. The method presented does not require the computation of all similarity values for the data. It is therefore very time efficient. When applied to the tissues (columns of the raw data gene expression matrix) a good separation of tumor form normal tissues is obtained, as described by the authors. The tumor rich cluster contains 35 tumor tissues and 3 normal tissues.

We performed a similar analysis using our algorithm. We used a publicly available subset of data, consisting of 2000 genes with the highest minimal intensity across the tissues. Figure 10 shows the results of clustering tissues using Pearson correlation based similarity function. Joining the tumor rich clusters yields a single cluster with 36 tumor and 3 normal tissues, similar to the clustering of [2]. Figure 11 depicts the original similarity matrix, and the same matrix with rows and columns permuted according to the order of the clusters produced by the algorithm.

In conclusion, the purpose of this section is to highlight the tissue differentiation (tumor vs normal) information that is contained in gene expression data. It demonstrates the usefulness of clustering techniques in learning more about the relationship of expression profiles to tissue types, including the generation of new hypotheses and the identification of directions for further studies. Compared to hierarchical methods, direct clustering has the advantage of automatically producing sub-clusters. It is not clear whether the different tumor rich clusters obtained here represent any biologically meaningful groups.



A B C

Figure 9: Clusters of the Let23 time course. A: cluster #11: seven out of the eight members are sperm proteins. The increase in expression occurring at the middle of the time course corresponds to the developmental stage at which such increase is expected. B: cluster #13 contains two copies of NADH dehydrogenase. C: cluster #17 contains two carboxypeptidase proteins (not color coded). Note the homogeneity of the expression patterns in each cluster.

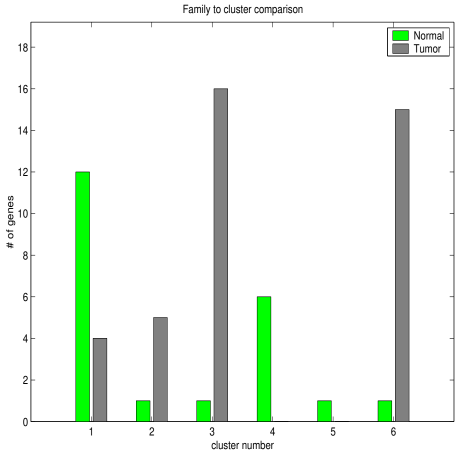


Figure 10: Distribution of tumor and normal tissues in the six clusters produced by the algorithm on a subset of the gene expression data of Alon et. al.. x axis: cluster number. y axis: number of tissues.

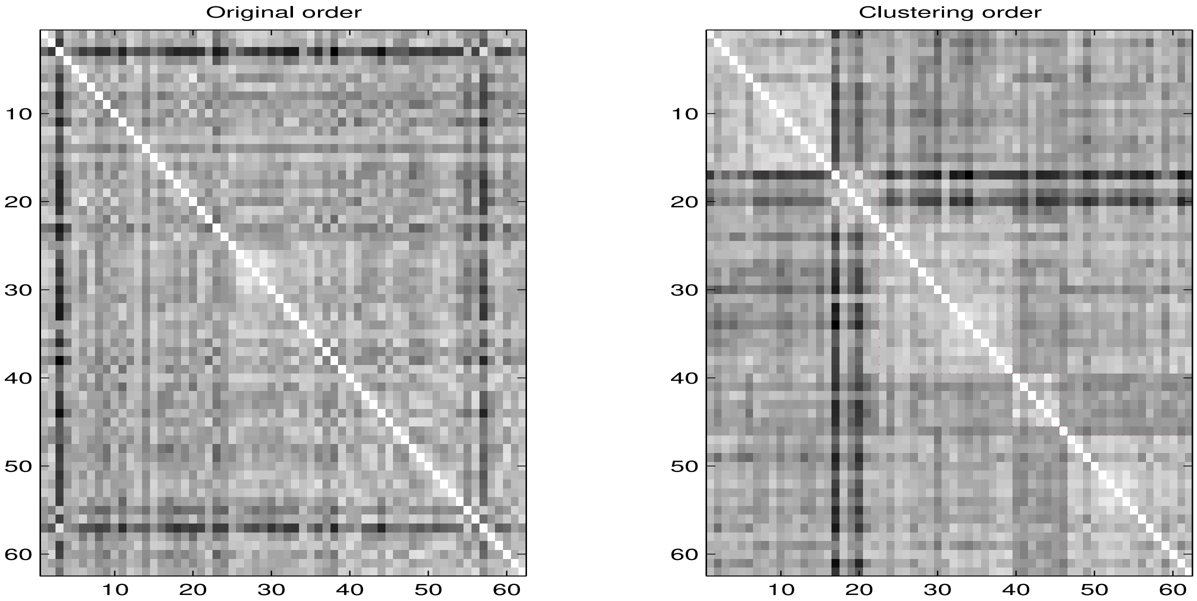


Figure 11: A visual representation of the similarity matrix for the tissue clustering data. Rows and columns correspond to tissues; The color of cell (i, j) corresponds to the similarity between tissues i and j, where lighter color represents bigger similarity. Left: The original matrix, order of tissues is arbitrary. Right: The same matrix, reordered according to the solution of the algorithm. Cluster order is same as in Figure 10, namely, clusters of sizes 12,6,17,6,1,16. Ordering of the tissues within each cluster is arbitrary. Observe that the clusters (except perhaps the smallest) can be now visually detected as lighter rectangles.

5 Discussion

The availability of very large quantities of gene expression data has the potential to revolutionize the way gene functions are discovered. A bird's eye view of thousands of genes simultaneously can shortcut the process to raising a solid hypothesis on a particular gene or network. Clustering has a central role in forming such hypotheses, as it can associate unknown genes with others of known function. The final validation of any such hypothesis will, of course, be done by "wet" biological work.

In this paper we have described two algorithms for clustering. One has a provably polynomial time and an asymptotic guarantee of correctness under a reasonable error model. The other is heuristic, with no formal proof of time complexity or even convergence. The heuristic has been implemented and tested on simulated noisy data and on real biological data. It manifested good performance, both in terms of time and in terms of solution quality. It remains an open question to prove complexity and solution quality properties for the heuristic algorithm.

The proposed algorithms have several advantages over other algorithms that have been exploited for clustering gene expression data: They form a non-hierarchical clustering, i.e., the proposed clusters are unrelated, and cluster boundaries are determined by the algorithm, without human intervention. Moreover, the number of clusters is determined by the algorithm, instead of being an constant given as input parameter to the program. (The affinity threshold parameter indirectly influences the cluster structure. In one variant of the heuristic, a procedure that optimally chooses that parameter was implemented.) The heuristic also has the advantage of being a variation on a well-understood algorithm with provable qualities. The choice of the preferred algorithm will depend on the application and the goals of the experimentor. It is of interest to perform a thorough comparison among the various gene expression analysis algorithms on simulated and real data, in order to determine which algorithm is preferable in which situation.

The implemented algorithm was shown to be quite fast, which allows iterated, interactive application of it by the user. A convenient user interface with several visualization tools was developed, and will be described in detail elsewhere. We intend to develop the program further and turn it into a robust working tool for gene expression analysis.

Acknowledgements

We thank Amir Dembo for useful comments on rigorously handling the stochastic model. We are grateful to Stuart Kim for access to the *C. elegans* data and for stimulating discussions on computational approaches to gene expression data interpretation. We thank Mel Kronick for valuable advice and information.

References

[1] N. Alon, M. Krivelevich, and B. Sudakov. Finding a large hidden clique in a random graph. In *Proc. Ninth Annual ACM-SIAM Sympoium on Discrete Algorithms (SODA 98)*, pages 594–598, San Francisco, California, 1998.

[2] U. Alon, N. Barkai, D. A. Notterman, G. Gish, S. Ybarra adn D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS*, 96:6745–6750, June 1999.

[3] A. Ben-Dor and Z. Yakhini. Clustering gene expression patterns. In *Proceedings of the Third International Conference on Computational Molecular Biology (RECOMB '99)*. ACM Press, 1999.

[4] A. P. Blanchard and L. Hood. Sequence to array: probing the genome's secrets. *Nature Biotechnology*, 14:1649, 1996.

[5] H. Chernoff. A measure for the asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. of Math. Stat.*, 23:493–509, 1952.

[6] A. Condon and R. M. Karp. Algorithms for graph bisection on the planted bisection model. personal communication, 1998.

[7] T. M. Cover and J. M. Thomas. *Elements of Information Theory*. John Wiley & Sons, London, 1991.

[8] R. Drmanac G. Lennon S. Drmanac I. Labat R. Crkvenjakov and H. Lehrach. Partial sequencing by oligohybridization: Concept and applications in genome analysis. In *Proceedings of the first international conference on electrophoresis supercomputing and the human genome Edited by C. Cantor and H. Lim*, pages 60–75, Singapore, 1991. World Scientific.

[9] A. Dembo and O. Zeitouni. *Large Deviations Techniques and Applications*. Springer-Verlag, New York, second edition, 1998.

[10] J. DeRisi, V. Iyer, and P. Brown. Exploring the metabolic genetic control of gene expression on a genomic scale. *Science*, 278:680–686, 1997.

[11] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley-interscience, New York, 1973.

[12] M. B. Eisen and P. O. Prown. DNA arrays for analysis of gene expression. In *Methods in Enzymology, Vol. 303*, pages 179–205. 1999.

[13] M. B. Eisen, P. T. Spellman, P. O. Prown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[14] B. Everitt. *Cluster Analysis*. Edward Arnold, London, third edition, 1993.

[15] J. Felsenstein, 1993. PHYLIP (Phylogeny Inference Package) version 3.5c, Univ. of Washington, Seattle.

[16] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical programming*, 79:191–215, 1997.

[17] J.A. Hartigan. *Clustering algorithms*. John Wiley and sons, 1975.

[18] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, and R. Shamir. An algorithm for clustering cDNAs for gene expression analysis using short oligonucleotide fingerprints. In *Proceedings Third International Symposium on Computational Molecular Biology (RECOMB 99)*, pages 188–197. ACM Press, 1999.

[19] J. Khan, R. Simon, M. Bittner, Y. Chen, S. B. Leighton, T. Pohida, P. D. Smith, Y. Jiang, G. C. Gooden, J. M. Trent, and P. S. Meltzer. Gene expression profiling of alveolar rhabdomyosarcoma with cDNA microarrays. *Cancer Research*, 58(22):5009–5013, 1998.

[20] K. R. Khrapko, A. Khorlin, I. B. Ivanov, B. K. Chernov, Y. P. Lysov, S. Vasilenko, V. Floreny'ev, and Mirzabekov. Hybridization of DNA with oligonucleotides immobilized in gel: A convenient method for detecting single base substitutions. *Molecular Biology*, 25(3):581–591, 1991.

[21] S. Kim. Department of Developmental Biology, Stanform University, `http://cmgm.stanford.edu/~kimlab/`.

[22] L. Kučera. Expected complexity of graph partitioning problems. *Discrete Applied Math*, 57:193–212, 1995.

[23] G.S. Lennon and H. Lehrach. Hybridization analysis of arrayed cDNA libraries. *Trends Genet*, 7:60–75, 1991.

[24] C. Y. Lin, K. H. Hahnenberger, M. T. Cronin, D. Lee, N. M. Sampas, and R. Kanemoto. A method for genotyping cyp2d6 and cyp2c19 using genechip probe array hybridization. In *ISSX Meeting*, 1996.

[25] D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown. DNA expression monitoring by hybridization to high density oligonucleotide arrays. *Nature Biotechnology*, 14:1675–1680, DEC 1996.

[26] Y. Lysov, A. Chernyi, A. Balaev, F. Gnuchev, K. Beattie, and A. Mirzabekov. DNA sequencing by contiguous stacking hybridization on modified oligonucleotide matrices. *Molecular Biology*, 29(1):62–66, 1995.

[27] B. Mirkin. *Mathematical Classification and Clustering.* Kluwer academic publishers, 1996.

[28] P. A. Pevzner, Y. Lysov, K. R. Khrapko, A. Belyavsky, V. Floreny'ev, and A. Mirzabekov. Improved chips for sequencing by hybridization. *J Biomolecular Str. Dyn.*, 9(2):399–410, 1991.

[29] John A. Rice. *Mathematical Statistics and Data Analysis.* Wadsworth, California, 2nd edition, 1995.

[30] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[31] R. Somogyi, X. Wen, W. Ma, and J. L. Barker. Developmental kinetics of GAD family mRNAs parallel neurogenesis in the rat spinal cord. *J. Neurosci*, 15(4):2575–2591, 1995.

[32] P. T. Spellman, G. Sherlock, et al. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9:3273–3297, 1998.

[33] 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. *PNAS*, 96:2907–2912, 1999.

[34] V. E. Velculescu, et al. Characterization of the yeast transcriptome. *Cell*, 88:243–251, 1997.

[35] X. Wen, S. Fuhrman, G. S. Michaels, D. B. Carr, S. Smith, J. L. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development. *PNAS*, 95(1):334–339, 1998.