A Distributed, Parallel System for Large-Scale Structure Recognition in Gene Expression Data

Jens Ernst

Lehrstuhl für Effiziente Algorithmen,
Institut für Informatik,
Technische Universität München
ernstj@in.tum.de

Abstract. Due to the development of very high-throughput lab technology, known as *DNA microarrays*, it has become feasible for scientists to monitor the transcriptional activity of all known genes in many living organisms. Such assays are typically conducted repeatedly, along a timecourse or across a series of predefined experimental conditions, yielding a set of *expression profiles*. Arranging these into subsets, based on their pair-wise similarity, is known as *clustering*. Clusters of genes exhibiting similar expression behavior are often related in a biologically meaningful way, which is at the center of interest to research in functional genomics.

We present a distributed, parallel system based on spectral graph theory and numerical linear algebra that can solve this problem for datasets generated by the latest generation of microarrays, and at high levels of experimental noise. It allows us to process hundreds of thousands of expression profiles, thereby vastly increasing the current size limit for unsupervized clustering with full similarity information.

Keywords: computational biology, structure recognition, gene expression analysis, unsupervized clustering, spectral graph theory.

1 Introduction

Computational Biology has provided a great wealth of complex and challenging algorithmic problems for researchers in the field of combinatorial optimization to work on. Moreover, the development of high-throuput lab technology has brought about a massive increase in the rate at which experimental data is generated and has to be processed by suitable algorithms. This not only calls for a great deal of effort in optimizing these algorithms for efficiency, but also presents a natural motivation for exploiting their parallelism and for distributing work across a network of computers. In this paper we consider *unsupervized clustering* of gene expression profiles as a particular case of a structure recognition problem where input data is generated at an industrial, ever increasing rate and at great cost, while efficient processing is indispensible to handling an otherwise unmanageable amount of information.

The DNA of all organisms contains a number of relatively short regions which code for protein, the main chemical ingredient to life. Genes are DNA sequences composed of such regions. The process by which the information encoded in a gene is transcribed and protein molecules are produced by cellular machinery is known as *gene expression*. This process is highly dynamic and very complex in that there exist networks of genes that mutually promote or suppress each other's expression, particularly in response to environmental changes affecting the cell. While it is possible to sequence entire genomes and to predict the location of genes at a reasonably high level of reliability, many questions on gene function and interaction are still unanswered. A great deal of research is directed towards understanding gene expression on a gobal level, taking into account many – perhaps all – genes of a given organism in parallel. This gives rise to the term *functional genomics*. The technology of *DNA microarrays* [5] has made it possible to observe the levels of expression activity of a very large number of genes in a single experiment and has become a standard research tool. In simplified terms, short but sufficiently representative nucleic acid subsequences (*probes*) of genes are physically spotted or photolithographically synthesized on a glass substrate. Then, in a hybridization reaction, these probes attract and bind to fluorescently labeled counterparts of their respective sequences from, say, a tissue sample. Later, fluorescence scanning of the array allows quantitative inferences on the activity of each gene on the microarray in terms of its level of transcription, and hence its expression. Often, but not always, probes correspond to genes.

Typically, a series of multiple experiments using identical arrays is performed under different experimental conditions or along a timecourse, yielding a vector of expression scores for each probe. Vectors of this type are known as *expression profiles*. Let n be the number of probes and let m be the number of array experiments. Then the profiles can be written in the form of an $n \times m$ matrix of expression values. Typical values for n range up to $4 \cdot 10^6$, whereas $m \leq 50$ is usually a consequence of array cost and other practical limitations. The goal of clustering in this context is to partition the set of probes, represented by their corresponding expression profiles, into subsets (or *clusters*) in such a way that members of the same cluster exhibit highly similar expression behavior (or *coexpression*), while similarity between clusters should be low. To formally capture this, we assume to have a similarity measure $s : \mathbf{R}^m \times \mathbf{R}^m \longrightarrow [0,1]$ which maps each pair of expression profiles to a similarity score. A score of 1 indicates perfect similarity. This function is symmetric but not necessarily metric. In practice, Pearson correlation and measures based on L_2 distances are commonly applied.

The input for the clustering problem is hence a symmetric $n \times n$ *similarity matrix* A containing values between 0 and 1 and can be thought of as the edge weight matrix of an undirected graph. The diagonal elements all have a value of 1. The problem hence consists in partitioning the index set $\{1, 2, \ldots, n\}$ into subsets representing the clusters. But the partitioning problem is complicated by the fact that the number of clusters is not known a priori and that microarray technology is extremely prone to experimental error, leading to incorrect expression values, which translates into false positive and false negative similarity scores in A.

Extensive work has been done on clustering algorithms for gene expression data since microarray technology was introduced. For a comprehensive survey of the most popular methods, the reader is referred to [3,6,8]. The main limitation shared by all existing algorithms is on the maximum feasible value of n. The author is not aware of any current commercial or academic software system that can handle more than some tens of thousands of probes. Our system is designed to improve this limit on n by an order of magnitude. We shall restate the clustering problem more formally in the following Section. In Section 3 we briefly derive the sequential *SR-algorithm* ("Spectral Reduction") which is well suited for this clustering problem. We then propose a set of techniques for parallelizing and distributing this algorithm in Section 4. The final Section presents experimental results to demonstrate the quality and performance of the method.

2 The Clustering Problem

The following graph theoretical framework shall henceforth be used for modeling the input data to be processed by clustering algorithms.

Definition 1. *Given a set $S = \{X_1, \dots, X_n\}$ of expression profiles $X_i \in \mathbf{R}^m$ and a simliarity measure $s : \mathbf{R}^m \times \mathbf{R}^m \to \mathbf{R}$, the associated* similarity graph *for S is a complete, undirected graph $G = (V, E, w)$ with self-loops, where $|V| = n$ and $E = \big\{\{v, v'\} : v, v' \in V\big\}$. For $1 \leq i, j \leq n$, each vertex v_i is bijectively associated with one expression profile X_i and each edge $\{v_i, v_j\}$ is weighted with the score of the similarity $w(v_i, v_j) := w(v_j, v_i) := s(X_i, X_j)$ between X_i and X_j.*

Within this formal framework we now define a data model for our clustering problem. We assume that an unknown cluster structure exists in the input data and is perturbed by some random noise, modeling experimental data error.

Definition 2. *A* cluster graph *is an edge-weighted, complete, undirected graph $G_0 = (V, E, w_0)$ with self-loops whose vertex set V can be partitioned into K clusters C_k such that $w_0(v_i, v_j) = 1$ for all edges $\{v_i, v_j\}$ within the same cluster C_k, $1 \leq k \leq K$, and $w_0(v_i, v_j) = 0$ for edges $\{v_i, v_j\}$ connecting different clusters. Let p^{int} and p^{ext} be two different distributions on the interval $[0, 1]$ with respective expectations μ^{int} and μ^{ext}. A (p^{int}, p^{ext})–*perturbation*, applied to G_0, yields a randomly weighted* perturbed cluster graph *$G = (V, E, w)$ where each edge $\{v_i, v_j\}$ is independently assigned a weight $w(v_i, v_j) = w(v_j, v_i)$ with respect to distribution p^{int}, if $w_0(v_i, v_j) = 1$ and with respect to p^{ext}, if $w_0(v_i, v_j) = 0$.*

This allows us to formally state the clustering problem: Let $G = (V, E, w)$ be the result of some (p^{int}, p^{ext})-perturbation with $\mu^{int} \neq \mu^{ext}$, applied to a fixed but unknown cluster graph $G_0 = (V, E, w_0)$ whose vertex set consists of the clusters $C_1, \dots, C_K$. Let A_0 be the unknown edge weight matrix of G_0. Given G or its edge weight matrix A, our task is to reconstruct G_0 by partitioning V into the K original clusters C_k. The value of K is not necessarily known. It is assumed that each cluster C_k is of size $c_k n$ for some constant $c_k > 0$. The vector $(c_1, \dots, c_K)$ is called *cluster structure* of G. Also, we assume that $\{c_1, \dots, c_K\} = \{c_1, \dots, c_r\}$ where $c_1, \dots, c_r$ are each unique and m_i is the multiplicity of c_i for $1 \leq i \leq r$.

3 Spectral Properties of Perturbed Cluster Graphs

In this Section we survey some fundamental mathematical properties of the input similarity matrix A, which will help us in deriving an algorithm for the above clustering problem. To this end, we first consider the unperturbed matrix A_0 and then examine the effect of random noise.

Lemma 1. *The spectrum of A_0 consists of* 0 *and the set $\{c_1 n, c_2 n, \ldots, c_r n\}$. Each eigenvalue $c_i n$ has multiplicity m_i, and its associated eigenspace is spanned orthogonally by characteristic vectors of the m_i clusters of size $c_i n$.*

By multiplying A_0 with the characteristic vectors of the individual clusters, one can immediately verify that the characteristic vectors belonging to the K individual clusters are indeed eigenvectors, with the respective clusters sizes $c_i n$ as associated eigenvalues. Orthonormality is also obvious. As Rank $A_0 = K$, it follows that no additional non-zero eigenvalues exist.

Note that this immediately suggests an algorithm for identifying clusters. A simple rotation of any set of dominant eigenvectors of A_0 yields a set of characteristic vectors indicating cluster membership. A different way of thinking of the situation is to consider in a column-wise fashion a matrix $(K \times n)$-matrix Z_0 whose rows contain a set of mutually orthonormal vectors spanning the dominant eigenspaces. Each column, representing a vertex, corresponds to a point in $\mathbf{R}^K$, and two vertices belong to the same cluster if and only if they correspond to the same point. So instead of solving the trivial clustering problem for G_0 by looking for connected components of edges with weight 1, one can obtain the desired partition by examining the dominant eigenvectors of A_0.

Next, we can ask about the effects of (p^{int}, p^{ext})-perturbation which transforms A_0 into A. It is a well-known fact of matrix theory that the spectrum of a symmetric matrix is relatively insensitive to component-wise perturbation – in fact, eigenvalues are said to be *perfectly conditioned.* We will show that they are stable enough to still reflect the number of clusters, K. The eigenspaces, on the other hand, will be shown to be sufficiently stable to allow the reconstruction of the unperturbed cluster structure with high probability. The following theorem summarizes these results. Due to page restrictions, we omit the proof here and refer the reader to [1] where we provide this proof in great detail.

Theorem 1. *With probability $1 - o(1)$, matrix A has the following properties: The spectrum of A consists of K dominant eigenvalues of magnitude $\Theta(n)$ and $n - K$ eigenvalues of magnitude $O(n^{\frac{1}{2}})$, counting multiplicity. The eigenspaces associated with the dominant eigenvalues are spanned by vectors which are constant within the index sets associated with the individual clusters, up to a component-wise variation of $o(n^{-\frac{1}{2}})$. Let Z be a $(K \times n)$-matrix whose rows contain a set of mutually orthonormal vectors spanning the dominant eigenspaces. Then the columns z_t of Z $(1 \leq t \leq n)$ constitute n points in $\mathbf{R}^K$, and two columns t, t' satisfy $\|z_t - z_{t'}\|_\infty = o(n^{-\frac{1}{2}})$ if vertices v_t and $v_{t'}$ belong to the same cluster. Otherwise, it holds that $\|z_t - z_{t'}\|_\infty = \Omega(n^{-\frac{1}{2}})$.*

Algorithm 1. SR-algorithm

Input: *(p^{int}, p^{ext})-perturbed cluster graph $G = (V, E, w)$*
Output: *Adjacency matrix of G_0*

```
(i)   $A :=$ edge weight matrix of $G$;
(ii)  $\gamma := |V|^{2/3}$;
(iii) $K$:=number of eigenvalues $\lambda \geq \gamma$;
(iv)  $\{x_1, x_2, \ldots, x_K\}$ :=orthonormal set of eigenvectors associated with eigen-
      values $> \gamma$;
(v)   $Z := [x_1, x_2, \ldots, x_K]^{\mathrm{T}}$;
(vi)  Identify the $K$-clusters defined by the column vectors of $Z$;
(vii) Construct adjacency matrix $A_0$ based on the $K$-clusters;
```

As a consequence, even in the presence of (p^{int}, p^{ext})-perturbation, the number of clusters can be obtained by counting the number of dominant eigenvalues. Any threshold γ satisfying $\gamma = o(n)$ and $\gamma = \omega(n^{\frac{1}{2}})$, can be used to distinguish them from the others. The columns z_t of Z define n points in $\mathbf{R}^K$ which form K groups named *K-clusters*. For sufficiently large n and with probability $1 - o(1)$, the K-clusters are disjoint and grow arbitrarily tight as n grows. The task of assigning each vertex to its cluster can be accomplished by identifying the K-clusters in $\mathbf{R}^K$. This constitutes a Euclidean clustering problem and can be solved by applying the K-means algorithm, a linear-time farthest-first method or thresholding. This leads us to the *Spectral Reduction* (*SR*) Algorithm 1.

A great advantage of the SR-algorithm over direct Euclidean clustering on the expression vectors is that it does not rely on a particular choice of the similarity measure. Regardless of how s is defined, the structure recognition problem is *reduced* to a simple Euclidean clustering problem, and the correct number of clusters is obtained as a by-product. For instance, one could envision designing a custom similarity measure such as the absolute value of Pearson Correlation (in order to define both, exact correlation and exact anti-correlation as maximal relatedness between expression profiles). Moreover, similarity matrix A need not even stem from a gene expression data set. Even if the biological entities whose similarity is described by the matrix do not have a vector representation at all, the clustering problem is transformed into a Euclidean one, and according vector representations in K-space are generated for these entities.

The only remaining issue to be addressed is how to compute an orthonormal basis of the dominant eigenspaces. It is not necessary to compute the entire eigen decomposition of matrix A as only a few dominant eigenvectors are needed. Instead we resort to an iterative numerical method based on Krylov subspaces and the well-known Rayleigh-Ritz procedure. Algorithm 2 gives a sketch of the method. Here we apply it as a black box and refer the reader to [2] for a detailed treatment of the theory behind it and [4] for implementation hints.

Algorithm 2. Compute eigenpairs

Input: $A \in \mathbf{R}^{n \times n}$
Output: *Approximate dominant eigenpairs* $(\widetilde{\lambda}_i, \widetilde{x}_i)$

```
(i)   Choose $v_1 \in \mathbf{R}^n$ with $\|v_1\| = 1$;
(ii)  $\beta_1 := 0$; $v_0 := \mathbf{0}$;
(iii) for $j = 1, 2, \ldots$ do
          $w_j := Av_j - \beta_j v_{j-1}$;
          $\alpha_j := \langle w_j, v_j \rangle$;
          $w_j := w_j - \alpha_j v_j$;
          $\beta_{j+1} := \|w_j\|$;
          $v_{j+1} := \frac{1}{\beta_{j+1}} w_j$;
          Compute $(\widetilde{\lambda}_i^{(j)}, \widetilde{x}_i^{(j)})$, $1 \le i \le j$ by the Rayleigh-Ritz procedure
          if $|\widetilde{\lambda}_j^{(j)}| < n^{\frac{2}{3}}$ then stop fi;
      od
```

4 The Parallel and Distributed SR-Algorithm

To identify the parts of the SR-Algorithm that can benefit the most from parallelization, let us first examine its complexity, assuming that similarity matrix A is given as input. Identifying the K tightly concentrated clusters of the columns of Z in Euclidean K-space requires no more than a simple linear-time computation, and storing matrix Z requires only linear space. While this could be parallelized in a rather straightforward way, the gain would be negligible. Computing a set of dominant eigenvalues of A, however, costs $\Omega(n^2)$ time and space. This severely limits the maximum feasible value of n. In the scenario of a typical uniprocessor with 4GB of RAM and without use secondary storage, we find this limit to be around $n \approx 40,000$. The processing time in this case is still within the range of minutes. As an aside, a limitation of n to roughly $40,000$ is also seen in all other academic and commercial clustering systems that the author is aware of. This leads us to focus our efforts on parallelizing Algorithm 2.

Notice that all quadratic-time operations within this procedure are applications of matrix A as an operator to given column vectors. Therefore, the best speedup can be expected from parallelizing all matrix-vector multiplications. Notice further that A is not involved in any other operations. This allows us to distribute A among multiple machines (and their RAM), thereby increasing the maximum permissible n. As a third essential observation, note that for all practical purposes, the number of iterations of Algorithm 2, and hence the overall number of matrix-vector multiplications, is a very small number $<< n$.[1]

[1] In [1] we show that the number of iterations is bounded by $O(\log n)$, and we give perturbation theoretical arguments suggesting that it is even a constant independent of n, although a formal proof of this conjecture is not yet completed.

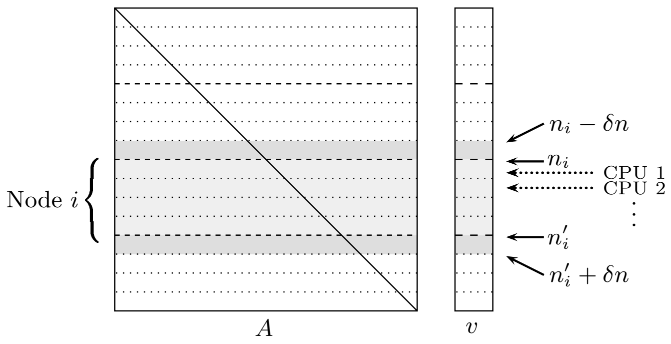

Fig. 1. Decomposition of matrix A and vector v for parallel multiplication

This last observation means that storing A and multiplying A with a series of vectors can be performed on a distributed-memory architecture consisting of machines that only communicate via a network protocol: The resulting amount of communication is of magnitude $O(n)$ and therefore not a serious limitation.

We choose our target architecture to be a general network of multiprocessor machines (*nodes*). The individual machines in the network communicate by generic message passing, for instance using the standard TCP/IP protocol. The processors within one node communicate via shared memory. This covers most of the architectures commonly used for scientific computation today, including the special cases of networks of uniprocessors and single multiprocessors.

One node is designated to play the part of a *coordinator* (or master). The gene expression profiles are distributed by the coordinator among all other machines which compute horizontal bands of the similarity matrix A to be held in local storage. The number of rows (indices block $[n_i, \ldots, n'_i]$) that each node i should store is determined prior to the distribution process by a static load balancing step in which the RAM capacity and the node's overall performance (number of inner products of length-n-vectors per millisecond) are measured. The index range associated with each machine is chosen proportional to the measured performance. In addition to the core index block, δn extra rows preceding it and δn rows following it are also stored by each node for subsequent dynamic load balancing (see Figure 1). All work to be performed by an individual node is evenly distributed among all the processors of this node. This requires no additional synchronization because computations on different rows of A are independent.

After all matrix bands have been computed, the coordinator generates the first vector v to which A should be applied (see Algorithm 2) and distributes it to all other nodes. The nodes perform the inner products required for generating their associated index block of Av. Again, this work is shared within each node by all the processors that communicate via shared memory. Then the coordinator collects the result blocks from each node, assembles and uses Av to finish the current iteration of Algorithm 2. Note that this presents a synchronization point for all nodes. Should the load on an individual node change significantly during this process (due to use in a multi user environment), the other nodes may be caused to wait as a consequence. For this reason, the coordinator process measures the time required by the individual nodes and adjusts the index ranges proportionally to their performance for the next iteration, using the $2\delta n$ extra rows stored within each node, if necessary. In our experiments we use $\delta = 0.05$.

5 Experimental Results

In this Section we shall examine the quality of the cluster decomposition resulting from the application of the SR-algorithm to various types of input data, as well as the algorithm's performance on a current multiprocessor architecture. We assess the clustering quality based on synthetically generated similarity data consisting of some known, planted structure which was obscured by adding significant amounts of noise. This data complies with the data model introduced in Section 2. A cluster structure $(c_1, c_2, \ldots, c_K)$ is imposed, resulting in clusters of respective sizes $c_1 n, c_2 n, \ldots, c_K n$. Then, perturbation is modeled by the two distributions p^{int} and p^{ext}. We choose binary similarity scores and Bernoulli distributions, causing false positive similarity with some probability α and false negative scores with probability β. This is referred to as *(α, β)-perturbation.* After the clustering process, we quantify the quality of the resulting partition, using the *Matching coefficient* and the *Minkowski measure* defined as follows.

Definition 3. *Given the two matrices $A_0, A_1 \in \{0,1\}^{n\times n}$ (where A_0 represents the true solution and A_1 represents the solution to be evaluated), we define*

$$n_{k,l} := \left|\{(i,j) : 1 \leq i,j \leq n, [A_0]_{i,j} = k, [A_1]_{i,j} = l\}\right|$$

for $k,l \in \{0,1\}$. Based on these quantities, we define the following measures:

(i) Minkowski Measure $M_1 := \sqrt{\frac{n_{0,1}+n_{1,0}}{n_{1,1}+n_{1,0}}}$

(ii) Matching Coefficient $M_2 := \frac{n_{0,0}+n_{1,1}}{n_{0,0}+n_{0,1}+n_{1,0}+n_{1,1}}$

Note that $n_{1,0}$ is the number of *false negatives* in the solution, $n_{0,1}$ is the number of *false positives*, whereas $n_{0,0}$ and $n_{1,1}$ are the true negatives and positives, respectively. In the ideal case where $A_0 = A_1$, the measures given in the above definition have the respective values of 0 and 1. Only the Minkowski Measure can potentially assume values greater than 1.

The results of our experiments can be seen in Figures 2 and 3. For growing values of n, the situation rapidly improves due to tighter concentrations in the Chernoff-type random variables which are the key to the correctness of Theorem 1. In this sense the choice of $n = 1,000$ presents a worst-case scenario.

To measure the performance of our implementation, we used a machine consisting of 32 *nodes*, each containing four AMD Opteron 850 processors at 2.4GHz with 8GB of RAM running Linux. The nodes are interconnected by Gigabit Ethernet. For this analysis we rely on a data set generated for the analysis of cigarette smoking-induced changes in bronchial epithelia, and reversibility of effects when smoking is discontinued. This data set may provide insight to molecular events leading to chronic obstructive pulmonary disease (COPD) and lung cancer [7].

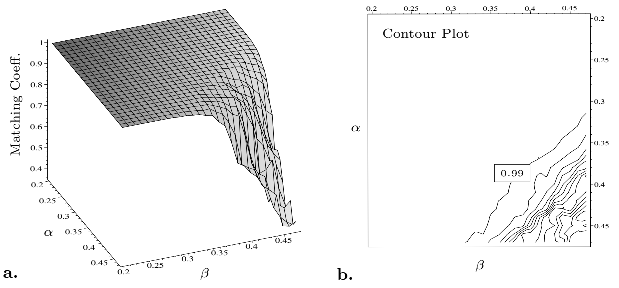

Fig. 2. a. The Matching Coefficient as a function of α and β for an (α, β)-perturbed cluster graph with relative cluster sizes $(0.1, 0.2, 0.3, 0.4)$ and $n = 1000$. **b.** Contour plot of the Matching Coefficient, showing parameter pairs with equal Matching scores.

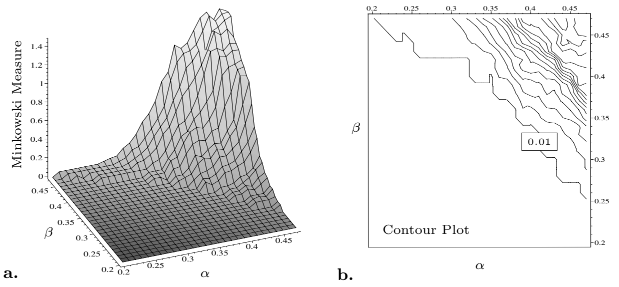

Fig. 3. a. Minkowski Measure as a function of α and β for an (α, β)-perturbed cluster graph with relative cluster sizes $(0.1, 0.2, 0.3, 0.4)$ and $n = 1000$. **b.** Contour plot of **a.**

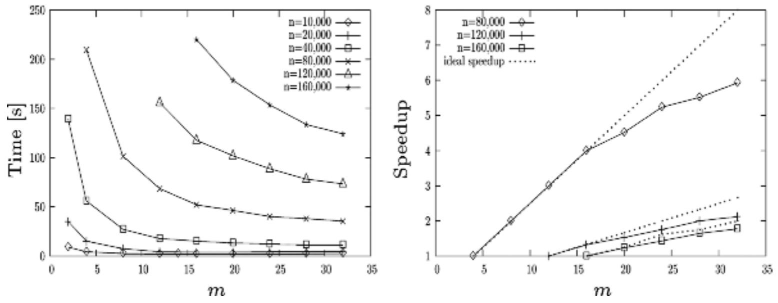

Fig. 4. a. Running times as a function of the number n of gene expression profiles and as a function of the number m of nodes. **b.** Speedup diagram for multiple input sizes

It was created using an Affymetrix GeneChip®Human Genome U133 microarray set HG-U133A. We conduct our cluster analysis in single-probe resolution, meaning that probes belonging to the same transcript of the same gene are considered separately rather than as components of a sum signal. In this experiment we generate 1,000 clusters, bypassing the automatic detection of the number of clusters, to simulate more realistically the way in which clustering algorithms are applied in practice to this type of expression data. We measure the running time as a function of the number of expression profiles and as a function of the number m of multiprocessors used in the computation (see Figure 4). For each value of n, m ranges between the minimum number of machines required to complete the run without resorting to secondary storage, and the maximum value of 32. This analysis shows that, as m is increased, the speedup initially is near-linear and then decreases somewhat, due to communication overhead. This should be expected because the amount of communication increases as a function of m.

Its ability to handle values of n on the order of $k \times 10^5$ with very moderate running time (e.g. $n = 160,000$ in less than 3 minutes) allows the parallel SR-Algorithm to process microarray data of the latest generation, e.g. Affymetrix Exon and Tiling Arrays®, without requiring a priori data filtering.

6 Conclusion

In this paper we have introduced a parallel, distributed algorithm for recovering cluster structures hidden in large and strongly noise-contaminated sets of gene expression data, originating from large-scale DNA microarray experiments. Along with the algorithm, we have introduced a formal data model within which its correctness can be proven. Moreover, we have described how to efficiently solve the numerical problems induced by the algorithm. Experimental evidence for both, its high noise-robustness and performance was presented. Further developments will include, but not be limited to, compression of matrix A to further increase the limit on parameter n. The software will soon be made accessible at `http://wwwmayr.informatik.tu-muenchen.de/personen/ernstj/`.

References

1. Ernst J. *Similarity-Based Clustering Algorithms for Gene Expression Profiles*, Dissertation, TU München, 2003
2. Gourlay A, Watson G. *Computational Methods for Matrix Eigenproblems*, John Wiley & Sons, New York, 1973
3. Daxin Jiang, Chun Tang, Aidong Zhang. *Cluster Analysis for Gene Expression Data: A Survey*, IEEE Transactions on Knowledge and Data Engineering, 2004; 16(11), 1370-86
4. Press W, Teukolsky S, Vetterling W, Flannery B. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edn., Cambridge University Press, 1992
5. Schena M, Shalon D, Davis RW, Brown PO. *Quantitative monitoring of gene expression patterns with a complementary DNA microarray*, Science, 1995; 270:467-470
6. Shamir R, Sharan R. *Algorithmic approaches to clustering gene expression data*, Current Topics in Computational Biology, 2002; 269-300
7. Spira A, Beane J, Shah V, Liu G, Schembri F, Yang X, Palma J, Brody JS. *Effects of cigarette smoke on the human airway epithelial cell transcriptome*, Proc Natl Acad Sci US, 2004; 101(27):10143-8
8. Valafar F. *Pattern Recognition Techniques in Microarray Data:A Survey*, Special Issue of Annals of New York, Techniques in Bioinformatics and Medical Informatics, 2002; 980:41-64