

An Algorithm for Clustering cDNAs for Gene Expression Analysis

Erez Hartuv[¶] Armin Schmitt* Jörg Lange* Sebastian Meier-Ewert* Hans Lehrach*
Ron Shamir^{||}

Abstract

We have developed a novel algorithm for cluster analysis that is based on graph theoretic techniques. A similarity graph is defined and clusters in that graph correspond to highly connected subgraphs. A polynomial algorithm to compute them efficiently is presented. Our algorithm produces a clustering with some provably good properties.

The application that motivated this study was gene expression analysis, where a collection of cDNAs must be clustered based on their oligonucleotide fingerprints. The algorithm has been tested intensively on simulated libraries and was shown to outperform extant methods. It demonstrated robustness to high noise levels. In a blind test on real cDNA fingerprint data the algorithm obtained very good results. Utilizing the results of the algorithm would have saved over 70% of the cDNA sequencing cost on that data set.

1 Introduction

Cluster analysis seeks grouping of data elements into subsets, so that elements in the same subset are in some sense more closely associated with each other. The process of generating the subsets is called *clustering*, and the resulting subsets are called *clusters*. The association level is deter-

[¶]Department of Computer Science, Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv 69978 ISRAEL. erez@math.tau.ac.il URL: <http://www.math.tau.ac.il/~erez>

^{||}Department of Computer Science, Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv 69978 ISRAEL. shamir@math.tau.ac.il URL: <http://www.math.tau.ac.il/~shamir>. Supported in part by a grant from the Ministry of Science and Technology, Israel. Parts of this work were performed while this author was on sabbatical at the Department of Computer Science and Engineering, University of Washington, Seattle.

*Max Planck Institute for molecular genetics, Ihnestrasse 73, D-14195 Berlin, Germany

mined either by a set of features of each element, or by similarity values between pairs of elements. These values often originate from very noisy experimental measurements, which makes solving the problem well a hard challenge.

We present here a new clustering algorithm, based on a graph theoretic approach. A similarity graph is defined and clusters in that graph correspond to highly connected subgraphs, which are efficiently computable using network flow techniques. Our algorithm produces solutions with some provably good properties.

The application that motivated this study was gene expression analysis. One of the central goals in molecular biology and genetics is to elucidate the function of genes. By observing the level of expression of a gene in different phases in the organism life cycle, or in different disease stages, one can get important clues to understanding the gene's role. A common way to do this is by extracting cDNAs in large quantities from the tissue and then measuring the amount of cDNA of each gene in the sample. As the cDNAs are picked at random, the abundance of cDNAs extracted indicates the relative expression level of their genes.

Out of about 100,000 different genes that are present in human DNA, the number of different genes active in a human cell at any time is over 10,000 [12]. The relative abundance of cDNAs of different genes may vary by a factor of 10,000. This clearly implies that the size of the sample of cDNAs that must be extracted from a cell in order to obtain adequate representation of low-abundance genes must be 100,000 or more.

Sequencing all 100,000 cDNAs in a sample is slow, wasteful and prohibitively expensive. An alternative method was proposed about a decade ago [14, 3, 31, 4, 28, 22, 24]. It is based on spotting the cDNAs on high density filters (about 31,000 different cDNAs can be spotted currently in duplicates on one filter [28]). Short synthetic DNA sequences, typically 7-12 bases long, called *oligonucleotides* or *oligos* for short, can be generated in the laboratory. When many identical oligos, tagged with fluorescent dye or radioactive label, are put in touch with the filter in proper conditions, the oligos *hybridize* (attach) to all cDNAs that contain a DNA sequence complementary to that of the oligo. By in-

To appear in the Third International Conference on Computational Molecular Biology (RECOMB '99)

specting the filter one can detect to which of the cDNAs the oligo hybridized. Hence, ideally, the result of such an experiment is one 1/0 bit for each of the cDNAs.

By repeating the experiment with p different oligos, one obtains a p -long 0/1 vector for each cDNA, indicating which of the (complements of) oligo sequences are contained in each cDNA. This *fingerprint* vector, similar to a bar-code, identifies the cDNA, and thus cDNAs originating from the same gene should have identical fingerprints. Based on the fingerprints, one can devise computer algorithms to identify and cluster cDNAs originating from the same gene. As a result, ideally, only one cDNA will have to be sequenced from each cluster, and the size of each cluster will tell the abundance of its gene. This *oligo-fingerprinting* approach achieves high level of parallelism and is a very economical approach to expression analysis.

In practice, fingerprinting is error-prone, so an algorithm must be devised to handle this noise. Also, not all cDNAs originating from the same gene have the same length, so even in the errorless case their fingerprint will differ. Luckily, some cDNA extraction techniques guarantee that all cDNAs from the same gene will end at a common endpoint in the sequence [13], and therefore they all share a common sequence suffix*.

Alternative technologies such as DNA microchips have the advantage of being able to determine the expression levels of thousands of genes in parallel through a single hybridization. While this is clearly a very significant advantage over oligonucleotide fingerprinting, there are two caveats: 1) The sensitivity of the analysis is lower than with oligonucleotide fingerprinting and much larger amounts of tissue RNA are required for a single analysis. In some cases where tissues are rare, this is clearly limiting. 2) Only known genes can be analyzed by these methods. While the former point is being addressed increasingly successfully through technological developments, the latter is a severe drawback for the analysis of novel genes, or organisms with few identified genes. Large sequencing projects have successfully identified the majority of human genes, and will undoubtedly do the same for a limited number of model organisms (e.g. mouse, drosophila, c. elegans - which is now completed), it is highly unlikely however that the same resources will be made available for sequencing other important model genomes such as amphioxus, sea urchin and a number of plants and fungi. In order to analyse gene expression efficiently in these organisms, oligonucleotide fingerprinting currently offers the most effective strategy.

The new clustering algorithm given here has been tested intensively on simulated data and was shown to give good results even in the presence of relatively high noise levels. It was also shown to outperform a central algorithm that has been used for expression analysis. In a blind test on

*If internal priming may occur at polyA:polyT tracts, not all the cDNAs of the gene will have a common 3' endpoint. Empirically this phenomenon appears sufficiently rare to be ignored.

a real data set which was generated at the laboratory of Max Planck Institute of Molecular Genetics in Berlin, the algorithm achieved very good results. Using the algorithm for clustering that data would have saved over 70% of the cDNA sequencing cost.

Several graph theoretic approaches to cluster analysis have been suggested (see, e.g., [25, 17, 7]). Those include finding connected components, strongly connected components in directed graphs, cliques and maximal cliques. For a critique of these approaches see [16]. Two other approaches that are more similar to ours were proposed by Matula [18, 17, 16, 15] and by Wu and Leahy [32]. Both of these algorithms lack our important stopping criterion, with the ensuing provable results on the clustering quality. In particular these algorithms do not guarantee that clusters have diameter two, a key property of our clustering. For other limitations of these methods, see [9].

For the specific problem of clustering cDNA fingerprints, several approaches were suggested previously: Drmanac et al. [28] build clusters around connected components in the similarity graph. In that graph, vertices correspond to cDNAs and edges correspond to pairs whose similarity is above a threshold θ (see Section 2 for definitions.) Even with low false positives rate in the data, such algorithm would incorrectly connect true clusters. The way to avoid this is by increasing θ sharply, which causes the break of many clusters. Meyer-Ewert, Mott and Lehrach [22], build clusters around maximal cliques. First, all maximal cliques are computed and later maximal cliques with sufficiently large overlap are merged into a single cluster. Computing all maximal cliques is computationally intractable [6]. Moreover, a high false negative rate may break large clusters into many maximal cliques with a complicated and hard-to-detect overlap structure. Milosavljevic et. al. [24], build clusters using a greedy algorithm. In each step a new seed clone is chosen, and all clones that are sufficiently similar to the seed are added to its cluster and removed from the dataset. To avoid false separation of clusters, the algorithm is run twice, where before the second run an average fingerprint is computed for each cluster, and the fingerprints of all the clones in that cluster are replaced by it. Therefore, the second run of Greedy is a cluster merging method. Like most greedy approaches, this algorithm cannot handle well high noise levels, and the quality of its results are very sensitive to the starting point. Another algorithm, reported very recently in [13], combines ideas from the two previous methods. Our approach is completely different from all previous cDNA clustering methods. We shall show simulation results that demonstrate the superiority of our algorithm over the greedy algorithm.

2 The Highly Connected Subgraphs Algorithm

We start with some basic definitions on clustering (cf. [18, 8, 7]) and graph theory (cf. [2, 5, 17]). In the clustering problem, one has a *sample* L of n elements and an $n \times p$ real

data matrix of measurements D , which contains p measurements (or characteristics) on each of the n entities. From D one often forms an $n \times n$ real symmetric *similarity matrix* S describing pairwise *similarity values* between elements. The goal is to find disjoint subsets of L , which are called *clusters*, such that two criteria are satisfied: *Homogeneity*, i.e., entities in the same cluster are highly similar to each other; and *Separation*, i.e., entities from different clusters have low similarity to each other. Clusters of size 1 and n are called *trivial*. Elements not put into any cluster are collected into one group which is denoted S for *singletons*.

In the case of cDNA clustering, the sample L is a collection (or *library*) of n cDNA clones, and the data matrix is the *hybridization matrix* H , whose rows correspond to the clones and whose columns correspond to the oligos. $H_{i,j}$ is the intensity level of the hybridization of clone i with oligo j . We usually assume that the hybridization intensities were already translated to binary values. The i -th row, H_i , is the *fingerprint* of cDNA x_i . Since cDNAs which originate from the same gene have similar fingerprints, a good clustering algorithm should form a partition in which each cluster contains cDNAs originating from the same gene.

Given the similarity matrix S for a sample L , and a real value θ , the *similarity graph* [18, 25] with threshold value θ is the graph $G_\theta(L, E)$ where $E = \{(x_i x_j) | S_{i,j} \geq \theta\}$. The choice of θ will be discussed in a later section.

The *edge-connectivity* (or simply *connectivity*) $k(G)$ of a graph G is the minimum number of edges whose removal results in a disconnected graph. If $k(G) = l$ then G is called an l -*(edge)-connected graph*.

A *cut* (or cut set) in G is a set of edges whose removal disconnects the graph. A *minimum cut* is a cut with minimum number of edges. If C is a minimum cut set of a non-trivial graph G , then $|C| = k(G)$. Hence a k -connected graph is a nontrivial graph in which the size of a minimum cut is k .

The Basic Algorithm: Had the similarity graph represented the cluster structure perfectly, each cluster would have been a clique, as all members of a cluster are highly similar, and no two clusters would be connected by an edge, as elements from distinct clusters are supposed to be dissimilar. In reality, searching for cliques in the graph would fail in two ways: First, finding maximum cliques is computationally intractable [6]. Second, and more important, real data matrices (and cDNA hybridization matrices in particular) contain many errors. In terms of the similarity graph, false negative errors correspond to missing edges between vertices in the same cluster, and false positive errors correspond to extra edges between vertices in different clusters. In cDNA fingerprinting, errors in the hybridization data generate inexact fingerprints, leading in turn to errors in the similarity graph. That error rate is very high: With a choice of θ that fits our algorithm best, the false negative rate in the graph is above 50%, and the false positive rate is smaller but still significant. Our algorithm was designed to withstand high error rate and work in low-degree polynomial time.

A key definition for our approach is the following: A graph G with $n > 1$ vertices is called *highly connected* if $k(G) > \frac{n}{2}$. A *highly connected subgraph* (HCS) is an induced subgraph $H \subseteq G$, such that H is highly connected. Our algorithm identifies highly connected subgraphs as clusters. The basic algorithm is shown in Figure 1. We assume that procedure $\text{Mincut}(G)$ returns H, \overline{H} , and C , where C is a minimum cut set which separates G into the subgraphs H and \overline{H} .

```

HCS(  $G(V, E)$  )
  ( $H, \overline{H}, C$ )  $\leftarrow$  MINCUT( $G$ )
  if ( $|C| > \frac{|V|}{2}$ ) then return ( $G$ )
  else
    HCS( $H$ )
    HCS( $\overline{H}$ )

```

Figure 1: The basic HCS algorithm.

The running time of the basic HCS algorithm (Figure 1) is bounded by $2N \times f(n, m)$, where $f(n, m)$ is the time complexity of computing a minimum cut in a graph with n vertices and m edges, and N is the number of clusters. Typically $N \ll n$. Currently, the best time bound for $f(n, m)$ is $O(nm)$ [19, 26].

Properties of HCS clustering: We now present several properties of the solutions produced by HCS algorithm. These properties will imply that appropriateness of the solutions for cluster analysis. Due to lack of space the proofs are omitted. The reader is referred to [9] for full proofs and more details.

The *distance* $d(u, v)$ between two nodes u and v in G is the number of edges in a shortest path between them, if such path exists; otherwise $d(u, v) = \infty$. The *diameter* of a connected graph G is the longest distance between any two nodes in G . The following simple theorem shows that our clusters satisfy strong homogeneity:

Theorem 1 *The diameter of every highly connected graph is at most two.*

Lemma 2 *Let S be a set of edges forming a minimum cut in the graph $G = (V, E)$. Let H and \overline{H} be the induced subgraphs obtained by removing S from G , where $|V(\overline{H})| \leq |V(H)|$. If $|V(\overline{H})| > 1$ then $|S| \leq |V(\overline{H})|$, with equality only if \overline{H} is a clique.*

The lemma implies that if a minimum cut S in $G = (V, E)$ satisfies $|S| > \frac{|V|}{2}$ then S splits the graph into a single vertex $\{v\}$ and $G \setminus \{v\}$. This shows us that using a stronger stopping criterion in our algorithm (Figure 1), i.e., $|S| > \alpha > \frac{|V|}{2}$ will be detrimental for clustering: Any cut of value x , $\frac{|V|}{2} < x \leq \alpha$ separates only a singleton from the current graph.

Theorem 3 *Let S be a minimum cut in the graph $G = (V, E)$ where $|S| \leq \frac{|V|}{2}$. Let H and \overline{H} be the connected*

induced subgraphs obtained by removing S from G , where $|V(\overline{H})| \leq |V(H)|$. If $\text{diam}(G) \leq 2$ then (1) every vertex in \overline{H} is incident on S , and, moreover, (2) \overline{H} is a clique, and, if $|V(\overline{H})| > 1$ then $|V(\overline{H})| = |S|$.

It can be shown, using Theorem 3, that the union of two vertex sets split by any step of HCS is unlikely to induce a graph with diameter ≤ 2 if noise is random, and the vertex sets are not too small. Another aspect of the solution quality is summarized below:

Theorem 4 1. The number of edges in a highly connected subgraph is quadratic.
2. The number of edges removed by each iteration of the HCS algorithm is at most linear.

Improvements. I: Singleton Adoption: The basic HCS algorithm may leave certain vertices as unclustered singletons. Subsequently, each singleton is checked whether it fits into one of those clusters. For each singleton x we compute the number of neighbors it has in each cluster and in the singleton set S . If the maximum number of neighbors is sufficiently large, and is obtained by a cluster (and not by S), then x is added to that cluster. The process is repeated up to I times ($I=50$ was used in practice) in order to accommodate the changes in clusters as a result of previous adoptions.

In order to reduce erroneous adoptions in case the level of noise is high, we require also that the number of neighbors of x_i in the adopting cluster must exceed a lower bound ϕ . In our practical runs we used $\phi = 2$.

II: The low degree heuristic: When the input graph contains low degree vertices, one iteration of a minimum cut algorithm may simply separate a low degree vertex from the rest of the graph. This is computationally very expensive, not informative in terms of the clustering, and may happen many times if the graph is large and sparse. Removing low degree vertices from G_θ eliminates such iterations and significantly reduces the running time. The complete algorithm which incorporates this idea and the singleton adoption is shown in Figure 2. d_1, d_2, \dots, d_p is a decreasing sequence of integers given as external input to the algorithm. The complete HCS algorithm is shown in Figure 2. The ‘until’ loop adds theoretically an $O(n)$ factor to the complexity, as singletons split by the HCS algorithm are reconsidered in the next iteration, but in practice only very few (1-5) iterations are needed.

3 Simulation Results

We performed intensive tests of the algorithm on simulated data. The simulation process receives as input the following parameters: The number of genes in the experiment is N_{genes} . Gene i has C_i copies, so C_i is the true size of the cluster of gene i . Hence, the total number of clones in the simulation is $N = \sum_{i=1}^{N_{genes}} C_i$. C_a and C_b are the minimum and maximum length, respectively, of a cDNA. Clone

```

HCS_LOOP( $G(V, E)$ )
  for ( $i=1$  to  $p$ ) do
     $H \leftarrow G$ 
    repeatedly remove all vertices of degree  $< d_i$  from  $H$ 
    until (no new cluster is found) do
      HCS( $H$ )
      perform singletons adoption
      remove clustered vertices from  $H$ 
      remove clustered vertices from  $G$ 

```

Figure 2: The complete HCS algorithm.

lengths are generated according to a normal distribution with mean $\mu = \frac{C_a + C_b}{2}$ and standard deviation $\sigma = \frac{C_b - C_a}{6}$. The number of probes (i.e., oligos) is p . Probes are assumed to occur along a gene with Poisson distribution with rate λ . This assumption originally was suggested in [23] and was adopted by other researchers [1, 27, 20]. The probability that an oligo occurrence did not register (false negative probability) is α . False positive hybridizations are assumed to have Poisson distribution with rate β . All probe occurrences and error events are assumed to be independent. The result of the simulation is an $N \times p$ hybridization matrix H , in which $H_{ij} = 1$ if clone i hybridized with probe j , and $H_{ij} = -1$ otherwise. The similarity matrix is then computed by $S_{ij} = \sum_k H_{ik} H_{jk}$, which in turn is used to generate the similarity graph G_θ .

We note that the probabilistic assumptions are used only in order to generate the data for the simulations. They are not used in clustering the simulated data or the real data. The only assumption that the algorithm makes is that fingerprints of clones from the same cluster tend to have higher similarity values.

We used the following measure for the quality of a solution given the true clustering: One can describe a clustering of n elements by an $n \times n$ symmetric $(0, 1)$ matrix C , where $C_{ij} = 1$ iff i and j belong the same cluster. Given matrix representations of the true clustering T and any clustering C of the same data set, the *Minkowski measure* for the quality of C (cf. [29, 10]) is the normalized distance between the two matrices, $\frac{\|T-C\|}{\|T\|}$, where $\|T\| = \sqrt{\sum_i \sum_j T_{i,j}^2}$.

An example of the performance of the HCS clustering algorithm on simulated cDNA oligo fingerprint data with high level of noise is given in Figure 3. The clustering of the algorithm is remarkably close to perfect, and does much better than the greedy algorithm.

Figure 4 summarizes the results of systematic experiments with the HCS algorithm on simulated data, and studies the effect of changing the main parameters.

Figure 4.1 shows the performance of the algorithm for various sizes of problems. The results are consistently good and they improve as the problem size increases, as the effect of the smaller cluster diminishes. The figure also gives results of the greedy algorithm of Milosavljevic et al [24] on

the same problems. The HCS algorithm is consistently superior, and difference in quality is up to an order of magnitude. Moreover, the greedy algorithm seems to deteriorate as the problem size increases while the HCS algorithm improves. The standard deviation in the quality of the greedy solutions is substantially larger, due to the extreme dependence of that algorithm on the starting point.

Figure 4.2 describe the performance of the algorithm with different values for the threshold θ . As expected, low threshold values give a large number of false positive edges, and high thresholds give a large number of false negative edges. Both of these situations hinder the clustering algorithm. The optimal threshold value is about 45, but good results are achieved when $\theta \in [30, 70]$, so finding an optimal θ is not crucial.

The effective range of the number of probes (Figure 4.3) is 100-300. Increasing the false positive parameter β up to 0.001-0.0015 has negligible effect on the quality of the results. In contrast, α can be increased up to 0.4 or 0.5 with little effect. Beyond these values, quality decreases rapidly as the error parameter increases. These results are quite encouraging, as the error rate in real large-scale hybridization experiments is quite high, but it falls within the range giving high quality clustering results according to our experiments.

The simulation algorithm was written in MATLAB, while the clustering algorithm was written in C++ within the LEDA 3.4.1 environment [21]. (The minimum cut algorithm implemented in LEDA has an $O(nm + n^2 \log n)$ time complexity [30].) Average elapsed times on a 194 MHZ SGI challenge L machine with 32Kb instruction cache and 1024 main memory were about two minutes for the clustering. (Clustering of a 7800 elements simulation required six minutes).

4 Clustering real cDNA data

4.1 Results using the HCS algorithm

We present now results of a blind test of clustering real cDNA data. The input contained 2329 cDNAs, originating from 18 genes. These were part of a library of some 100,000 cDNAs prepared from purified peripheral blood monocytes. These were isolated and the cDNA synthesised at the Novartis Forschungsinstitut in Vienna, Austria. The true clustering, obtained by hybridization with long, unique sequences, is given in Table 1. Note the high variability in abundance of genes. 139 oligos were used for fingerprinting. The input for the test was a real-valued similarity matrix S for the cDNAs, with similarity values ranging from 3.42 to 139. G_θ was generated from S by setting $\theta = 110$. The results are shown in Figure 5. The algorithm found 17 clusters, and left 206 entities as singletons. The Minkowski score is 0.343. (Comparison with Greedy was impossible as it relies on hybridization data). In 14 out of the 17 clusters generated by the algorithm, over 92% of the entities belong to the same gene (true cluster). Therefore we call those clusters *almost*

Cluster	Size	Gene name
T_{18}	709	Ef1 alpha
T_{17}	285	clone 190B1
T_{16}	284	Cytochr c oxi
T_{15}	213	tubulin beta
T_{14}	187	40SRibo protS6
T_{13}	146	40SRibo protS3
T_{12}	108	40SRibo protS4
T_{11}	91	GAPDH
T_{10}	86	60SRibo protL4
T_9	67	Ef1 beta
T_8	43	Human calmodulin
T_7	39	heat shock cogKD71
T_6	32	heat shock cogKD90
T_5	14	Human TNF recep
T_4	12	Human AEBP1
T_3	10	clone 244D14
T_2	2	clone 241F17
T_1	1	Human anion ch

Table 1: True clusters and gene identities in the MPI dataset

pure. Note the high level of noise and the fact that the eight smallest clusters are virtually undetectable visually even in the correct cDNA order.

Although the clustering generated by the algorithm is not perfect, we argue that it is quite good and can serve as a useful tool in gene expression analysis. As the correct clusters are not known (in real experiments), and the main goal of cDNA clustering is to avoid repeated sequencing of cDNAs originating from the same gene, the following strategy can be used: Pick at random up to 10 cDNAs from each cluster and sequence them. If (say) 9 out of 10 give the same sequence, the cluster is with high probability almost pure and no more sequencing of its members is needed. Otherwise, all the members of the cluster are sequenced. A simple calculation shows that this strategy would have saved *over 70% of the sequencing cost* in this case.

5 Conclusions and future work

We have described a new algorithm for clustering and have tested it on real and simulated expression analysis data with very good results. We anticipate that given raw fingerprint data the quality of the solution will improve further, as we could then use cluster merging and also select the threshold value θ better. Also, experience with more real data examples would allow better tuning of the algorithm to their properties, as we have done for simulated data. The test reported here was the first run of the algorithm on real data where the solution is known.

Clearly our simulation results depend on the assumption that all probes have the same rate, which is not satisfied by randomly chosen probes on real gene sequences. However, as was shown in [20] this can be remedied by a judicious choice

of probes. The results on the real dataset also demonstrate that the effect of this problem is not large.

Additional improvements to the algorithm can be achieved by using faster minimum cut algorithm (e.g. [11]) and by attempting to find maximal highly connected subgraphs (e.g. using Matula's cohesiveness function [17]). Using a weighted minimum cut algorithm may also improve the results.

Further theoretical questions on the properties of the algorithm are of interest: Can we determine the optimal value of θ in the similarity graph? Our choice of θ was based on experience with simulated data, but determining a good value for datasets with unknown properties is a key question in both practice and theory. Can we determine a probabilistic threshold for the level of noise under which good (or perfect) clustering is guaranteed, with high probability? A comparison of our algorithm with classical clustering algorithms like k -means [8] may be interesting. One clear advantage of our algorithm is the fact that the number of clusters need not be prespecified. Comparison with classical hierarchical clustering algorithms is also of interest. Finally, although the algorithm we developed was tested in the context of gene expression, its use is not limited to this application. Testing the algorithm on real data arising in other applications is something we have already started doing.

Acknowledgment

We thank the anonymous RECOMB '99 referees for many insightful remarks.

References

- [1] F. Alizadeh, R. M. Karp, L. A. Newberg, and D. K. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. *Algorithmica*, 13:52–76, 1995.
- [2] F. Buckley and F. Harary. *Distance in graphs*. Addison-wesley, 1990.
- [3] 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.
- [4] S. Drmanac and R. Drmanac. Processing of cDNA and genomic kilobase-size clones for massive screening mapping and sequencing by hybridization. *Biotechniques*, 17:328–336, 1994.
- [5] S. Even. *Graph algorithms*. Computer Science Press, Potomac Maryland 20854, 1979.
- [6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.
- [7] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. *Mathematical programming*, 79:191–215, 1997.
- [8] J.A. Hartigan. *Clustering algorithms*. John Wiley and sons, 1975.
- [9] E. Hartuv. Cluster analysis by highly connected subgraphs with applications to cDNA clustering. Master's thesis, Department of Computer Science, Tel Aviv University, August 1998.
- [10] N. Jardine and R. Sibson. *Mathematical taxonomy*. John Wiley and sons, London, 1971.
- [11] D.R. Karger. Minimum cuts in near linear time. In *Proc. 28th annual ACM symposium on theory of computing*, pages 56–63, 1996.
- [12] L. Zhang W. Zhou V.E. Velculescu S.E. Kern R.H. Hruban S.R. Hamilton B. Vogelstein K.W. Kinzler. Gene expression profiles in normal and cancer cells. *Science*, 276:1268–1272, 1997.
- [13] S. Meier-Ewert J. Lange H. Gerst R. Herwig A. Schmitt J. Freund T. Elge R. Mott B. Herrmann H. Lehrach. Comparative gene expression profiling by oligonucleotide fingerprinting. *Nucleic Acids Research*, 26(9):2216–2223, 1998.
- [14] G.S. Lennon and H. Lehrach. Hybridization analysis of arrayed cDNA libraries. *Trends Genet*, 7:60–75, 1991.
- [15] D.W. Matula. The cohesive strength of graphs. In *The many facets of graph theory, Lecture notes in Mathematics No.110, Edited by G. Chartrand and S.F. Kapoor*, pages 215–221, Berlin, 1969. Springer-Verlag.
- [16] D.W. Matula. Cluster analysis via graph theoretic techniques. In R.C Mullin K.B Reid and D.P Roselle, editors, *Proc. Louisiana conference on combinatorics graph theory and computing*, pages 199–212. University of manitoba Winnipeg, 1970.
- [17] D.W. Matula. k -Components, clusters and slicings in graphs. *SIAM J. Appl. Math.*, 22(3):459–480, 1972.
- [18] D.W. Matula. Graph theoretic techniques for cluster analysis algorithms. In *Classification and clustering Edited By J. Van Ryzin*, pages 95–129. Academic Press, 1977.
- [19] D.W. Matula. Determining edge connectivity in $O(nm)$. In *Proceedings 28th IEEE symposium on foundations of computer science*, pages 249–251, 1987.
- [20] G. Mayraz and R. Shamir. Construction of physical maps from oligonucleotide fingerprints data. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB '99)*. to appear.
- [21] Mehlhorn and Naher. LEDA: A platform for combinatorial and geometric computing. *Communications of the ACM*, 38, 1995.
- [22] S. Meier-Ewert, R. Mott, and H. Lehrach. Gene identification by oligonucleotide fingerprinting – a pilot study. Technical report, MPI, 1995.
- [23] F. Michiels, A. G. Craig, G. Zehetner, G. P. Smith, and H. Lehrach. Molecular approaches to genome analysis: a strategy for the construction of ordered overlapping clone libraries. *CABIOS*, 3(3):203–210, 1987.
- [24] A. Milosavljevic, Z. Strezoska, M. Zeremski, D. Grujic, T. Paunesku, and R. Crkvenjakov. Clone clustering by hybridization. *Genomics*, 27:83–89, 1995.
- [25] B. Mirkin. *Mathematical classification and clustering*. Kluwer academic publishers, 1996.

- [26] H. Nagamochi and T. Ibaraki. Computing edge connectivity in multigraphs and capacitated graphs. *SIAM J. Disc. Math.*, 5:54–66, 1992.
- [27] D. M. Platt. and T. I. Dix. Comparison of clone-ordering algorithms used in physical mapping. *Genomics*, 40:490–492, 1997.
- [28] S. Drmanac N. A. Stavropoulos I. Labat J. Vonau B. Hauser M. B. Soares and R. Drmanac. Gene-representing cDNA clusters defined by hybridization of 57419 clones from infant brain libraries with short oligonucleotide probes. *Genomics*, 37:29–40, 1996.
- [29] R. R. Sokal. Clustering and classification: Background and current directions. In *Classification and clustering Edited By J. Van Ryzin*, pages 1–15. Academic Press, 1977.
- [30] M. Stoer and F. Wagner. A simple Min-Cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.
- [31] R. Drmanac S. Drmanac I. Labat R. Crkvenjakov A. Vicentic and A. Gemmell. Sequencing by hybridization: towards an automated sequencing of one milion M13 clones arrayed on membranes. *Electrophoresis*, 13:566–573, 1992.
- [32] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.

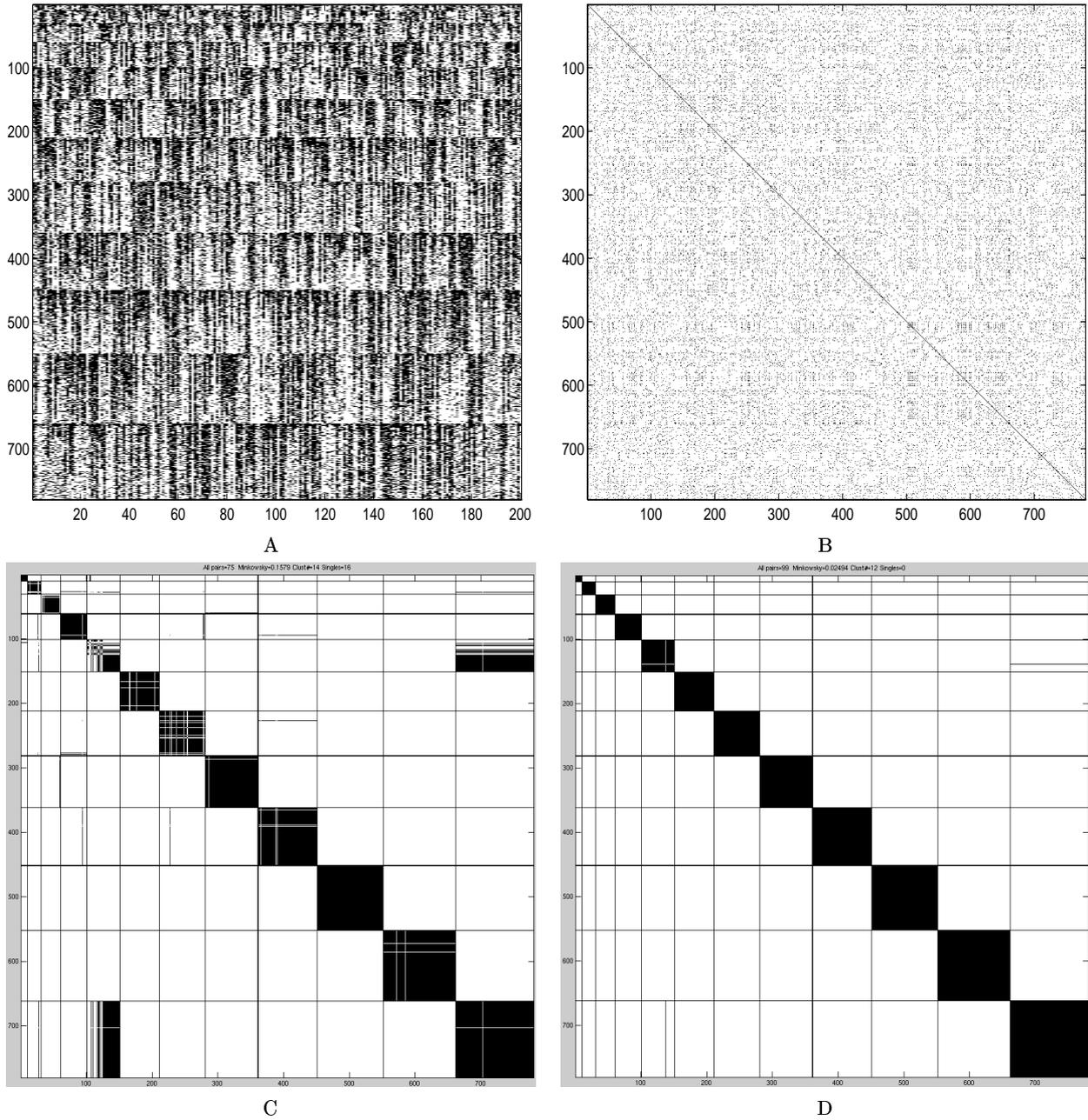


Figure 3: Example of results of HCS and Greedy clustering algorithms in high noise simulation. The fingerprint data consisted of 780 cDNAs from 12 genes, in clusters of sizes 10, 20, \dots , 120. The number of oligos is 200. $\beta = 0.0015$ so the expected rate of false positive hybridizations is 25%. $\alpha = 0.4$, so expected false negative hybridization rate is 40%. $C_a = 500$ bp $C_b = 2500$ bp. Poisson rate for oligo appearance is $\lambda = 0.005$. A: The hybridization fingerprints matrix H . Each of the 780 rows is a fingerprint vector of one cDNA. White denotes positive hybridization. B: The binarized similarity matrix. Position i, j is black iff $S_{ij} > 50$. Matrix coordinates are scrambled, as in realistic scenarios. C: Clustering solution generated by the greedy algorithm. Minkowski score is 1.32. To appreciate the results, cDNAs from the same true cluster appear consecutively, and black lines delineate borders between different clusters. Position i, j is black if the solution puts cDNAs i and j in the same putative cluster. D: Clustering solution generated by the HCS algorithm. Minkowski score is 0.209.

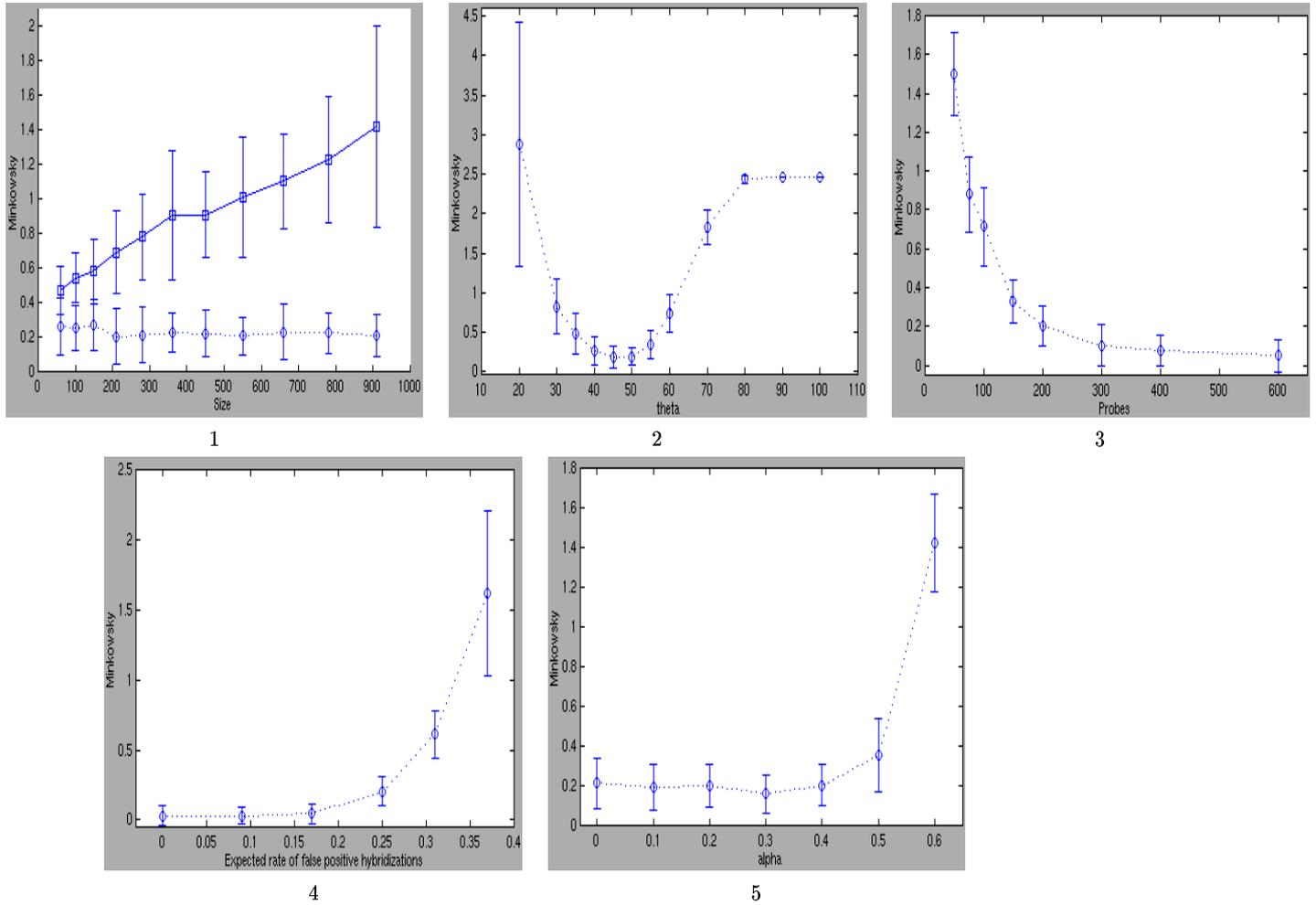


Figure 4: Impact of problem parameters on the performance of the algorithm, and comparison with the Greedy algorithm. Cluster structure: 450 elements in 9 clusters of sizes 10, 20, 30, ..., 90. The parameters used in the simulation were as reported in Figure 3. In each experiment one parameter value was changed while the rest were kept at the default values. Averages and standard deviations are based on 50 simulations per data point. All results are for the Minkowski score. 1: Impact of problem size and comparison to Greedy. Cluster structure is 10, 20, 30, ..., 130 with the sequence truncated after 3,4,..,13 elements. HCS: dotted line, Greedy: continuous line. 2: Impact of the threshold value. 3: Impact of the number of probes. 4: Impact of false positive rate. 5: Impact of false negative rate.

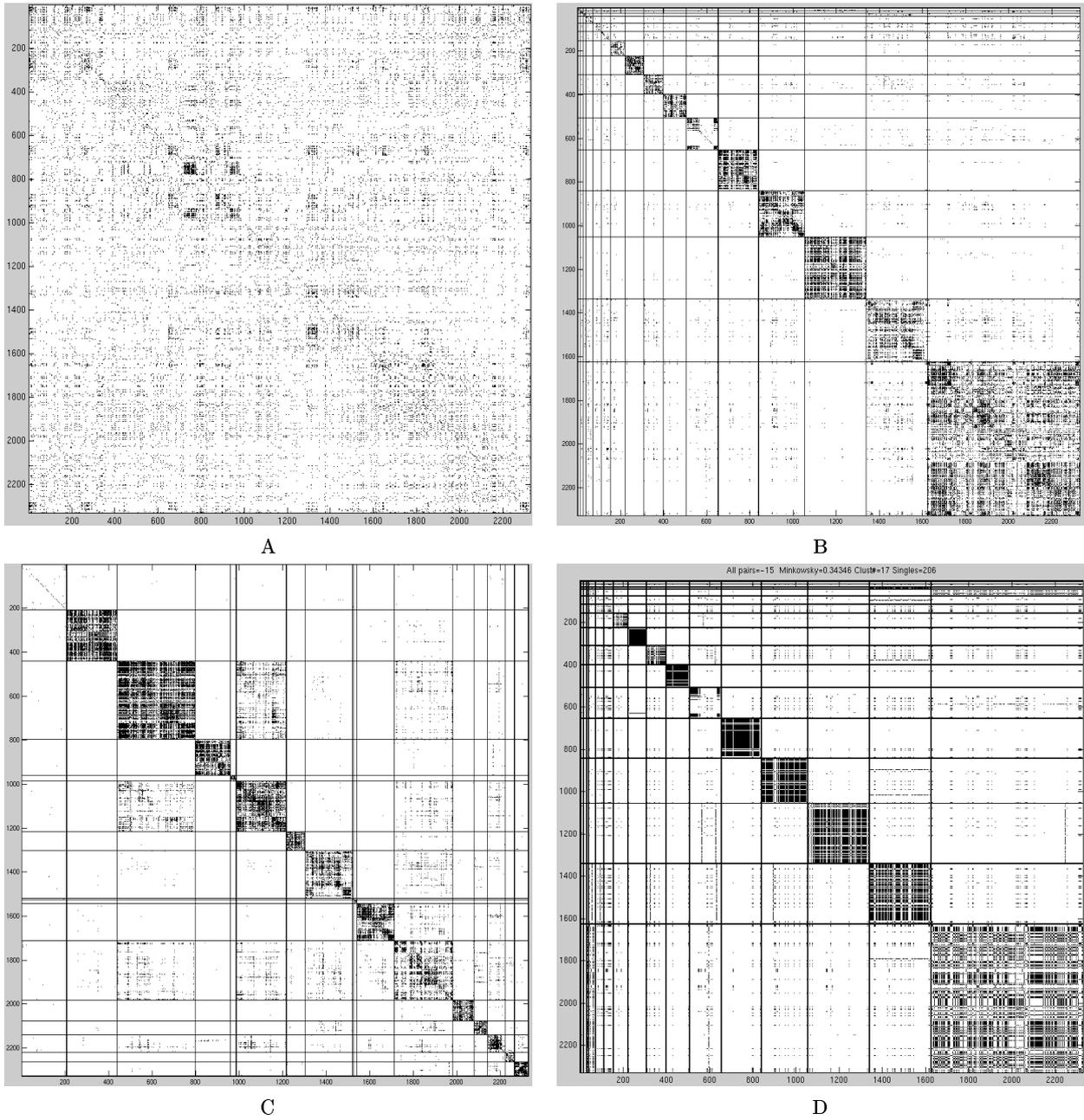


Figure 5: Clustering results on real cDNA data. A: The binarized similarity matrix. A black point appears at position (i, j) iff $S_{i,j} \geq 110$. B: Reordering of A according to the true clustering. cDNAs from the same true cluster appear consecutively, and black lines delineate borders between different clusters. C: Reordering of A according to the clustering produced by the HCS algorithm. Clusters appear in the order of detection. D: Comparison of the algorithmic solution and the true solution. Row and columns are ordered as in B. Position (i, j) is black iff the algorithm put i and j in the same cluster.