A Clustering Algorithm based on Graph Connectivity[0]

Erez Hartuv[¶] Ron Shamir[‖]

March 10, 1999, Revised December 1999

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 solution with some provably good properties and performs well on simulated and real data.

Keywords: Algorithms, Clustering, Minimum cut, Graph connectivity, diameter.

1 Introduction

Problem definition: Cluster analysis seeks grouping of elements into subsets based on similarity between pairs of elements. The goal is to find disjoint subsets, called *clusters*, such that two criteria are satisfied: *homogeneity*: elements in the same cluster are highly similar to each other; and *separation*: elements in different clusters have low similarity to each other. The process of generating the subsets is called *clustering*. The similarity level is usually determined by a set of features of each element. These often originate from noisy experimental measurements, and thus give inaccurate similarity values.

Motivation: Cluster analysis is a fundamental problem in experimental science, where one wishes to classify observations into groups or categories. It is an old problem with a history going back to Aristotle (cf. [5]). It has applications in biology, medicine, economics, psychology, astrophysics and numerous other fields. The application that motivated this study was gene expression in molecular biology.

Contribution of the paper: In this paper we present a new clustering algorithm. The approach presented here is graph theoretic. The similarity data is used to form a *similarity graph*

[¶]School 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

[‖]Corresponding author. School 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.

[0]Portions of this paper appeared in a preliminary version in the Proceedings of the Third International Conference on Computational Molecular Biology (RECOMB '99) [8].

in which vertices correspond to elements and edges connect elements with similarity values above some threshold. In that graph, clusters are highly connected subgraphs, defined as subgraphs whose edge connectivity exceeds half the number of vertices. Using minimum cut algorithms such subgraphs can be computed efficiently. We prove that the solution produced by our algorithm possesses several properties that are desirable for clustering.

Experimental Results: The algorithm has been implemented and tested intensively on gene expression simulated data and was shown to give good results even in the presence of relatively high noise levels, and to outperform a previous algorithm for that problem [17]. It has also obtained promising results in a blind test with experimental gene expression data [8, 9]. Further details can be found in [7].

Previous graph theoretic approaches: Due to its wide applicability, cluster analysis has been addressed by numerous authors in various disciplines in the past. The cluster separation and homogeneity goals described above can be interpreted in various ways for optimization. Numerous approaches exist depending on the specific objective function chosen (cf. [10, 1, 20, 6, 21, 18, 5]). We briefly review the approaches that are most related to our work. (Definitions and terminology will be given in Section 2.)

Matula [12, 13, 14, 15] was the first to observe the usefulness of high connectivity in similarity graphs to cluster analysis. Matula's approach is based on the *cohesiveness function*, defined for every vertex and edge of a graph G to be the maximum edge-connectivity of any subgraph containing that element. The components of the subgraph of G, obtained by deleting all elements in G of cohesiveness less than k, are precisely the maximal k-connected subgraphs of G. In [13] Matula suggested finding clusters by using a constant value k. The drawback in this approach is that different real clusters may have different connectivity values. Later [14] Matula suggested identifying as clusters maximal k-connected subgraphs (for any k) which do not contain a subcomponent with higher connectivity. This may cause the splitting of some real clusters that contain several highly cohesive parts.

Minimum cuts in capacitated similarity graphs were also used by Wu and Leahy [22]. The number of clusters K is assumed to be known for their algorithm. The $K-1$ smallest cuts in G are computed (e.g., using the Gomory-Hu algorithm [4]) and their removal produces a K-partition of the data. The resulting K-partition of G has two desirable properties: (1) it minimizes the largest inter-subgraph mincut among all possible K-partitions of G. (2) the maximum mincut between any pair of vertices in the same subgraph (intra-subgraph mincut) is always greater than or equal to the mincut between vertices in two different subgraphs (inter-subgraph mincut). In Section 5 we shall compare the two approaches with ours.

2 The HCS Algorithm

In this section we describe the Highly Connected Subgraphs (HCS) algorithm for cluster analysis. We first review some standard graph-theoretic definitions (cf. [3, 2]).

The *edge-connectivity* (or simply the *connectivity*) $k(G)$ of a graph G is the minimum number k of edges whose removal results in a disconnected graph. If $k(G) = l$ then G is called an *l-connected* graph. A *cut* in a graph is a set of edges whose removal disconnects the graph. A *minimum cut* (abbreviated mincut) is a cut with a minimum number of edges. Thus a cut S is a minimum cut of a non-trivial graph G iff $|S| = k(G)$. The *distance* $d(u, v)$ between vertices u and v in G is the minimum length of a path joining them, if such path exists; otherwise $d(u, v) = \infty$ (the *length* of a path is the number of edges in it). The *diameter* of a connected graph G, denoted $diam(G)$, is the longest distance between any two vertices in G. The *degree* of vertex v in a graph, denoted $deg(v)$, is the number of edges incident with it. The minimum degree of a vertex in G is denoted $\delta(G)$.

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 HCS algorithm is shown in Figure 1, and Figure 2 contains an example of its application. We assume that procedure MINCUT(G) returns H, $\overline{H}$, and C, where C is a minimum cut which separates G into the subgraphs H and $\overline{H}$. Procedure HCS returns a graph in case it identifies it as a cluster. Otherwise, it returns nothing, and subgraphs identified as clusters are returned by lower levels of the recursion. Single vertices are not considered clusters and are grouped into a *singletons* set $\mathcal{S}$. The collection of subgraphs returned when applying HCS on the original graph constitutes the overall solution.

```
HCS( G(V,E) )
begin
   (H, H̄, C) ← MINCUT(G)
   if G is highly connected
      then return (G)
   else
      HCS(H)
      HCS(H̄)
   end if
end
```

Figure 1: The HCS algorithm.

The running time of the HCS algorithm is bounded by $2N \times f(n, m)$, where N is the number of clusters found and $f(n, m)$ is the time complexity of computing a minimum cut in a graph with n vertices and m edges. Note that in many applications $N << n$. The current fastest deterministic algorithms for finding a minimum cut in an unweighted graph require $O(nm)$ steps, and are due to Matula [16] and Nagamochi and Ibaraki [19]. The fastest randomized algorithm is due to Karger and requires $O(m \log^3 n)$ time [11].

3 Properties of HCS Clustering.

In this section we prove some properties of the clusters produced by the HCS algorithm. These demonstrate the homogeneity and the separation of the solution.

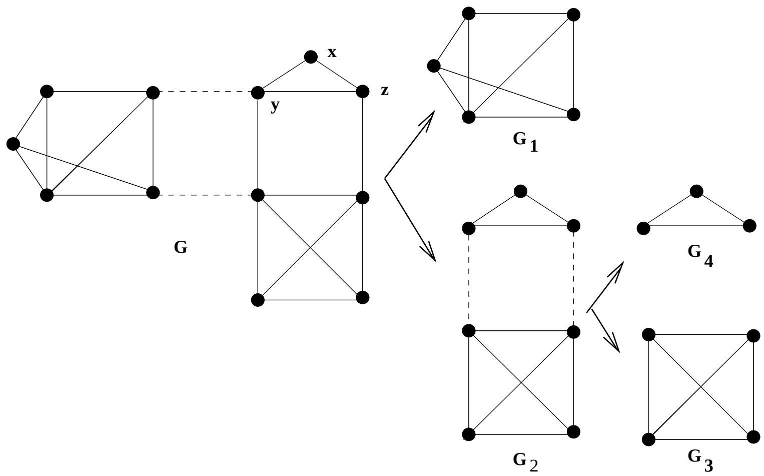

Figure 2: An example of applying the HCS algorithm to a graph. Minimum cut edges are denoted by broken lines.

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

Proof: When all the edges incident with a vertex of minimum degree are removed, a disconnected graph results. Therefore the edge connectivity of a graph is not greater than its minimum degree: $k(G) \leq \delta(G)$. If G is highly connected then $\frac{|V|}{2} < k(G) \leq \delta(G)$, so every vertex is adjacent to at least half of the vertices of G. Therefore every two vertices are at distance at most two, as they have a common neighbor. ■

Note that the theorem holds even if we allow the cardinality of the minimum cut in an HCS to be equal to $\frac{|V|}{2}$. Note also that the converse of Theorem 1 does not hold: Take for example the path on 3 vertices. In Theorem 3 below we give a characterization of the case where a graph has diameter 2 but is not highly connected. Using the characterization, we shall later argue that this situation is unlikely to occur in clustering real, noisy data.

Lemma 2 *If S is a minimum cut which splits the graph into two induced subgraphs, the smaller of which, $\overline{H}$, contains $k > 1$ vertices, then $|S| \leq k$, with equality only if $\overline{H}$ is a clique.*

Proof: Let $deg_H(x)$ denote the degree of vertex x in the induced subgraph H, and let $deg_S(x)$ be the number of edges in S that are incident on x. Since S is a minimum cut, for every $x \in \overline{H}$

$$deg_S(x) + deg_{\overline{H}}(x) \geq |S|$$

(Since otherwise $(\{x\}, V \backslash \{x\})$ would be a minimum cut). Summing over all vertices in $\overline{H}$ we get

$$\sum_{x \in \overline{H}} deg_S(x) + \sum_{x \in \overline{H}} deg_{\overline{H}}(x) \geq |S||V(\overline{H})|$$

or, equivalently

$$|S| + 2|E(\overline{H})| \geq |S||V(\overline{H})|$$

or

$$2|E(\overline{H})| \geq |S|(|V(\overline{H})| - 1)$$

Hence, if $|V(\overline{H})| > 1$,

$$|S| \leq \frac{2|E(\overline{H})|}{|V(\overline{H})| - 1} \leq \frac{2\frac{|V(\overline{H})|(|V(\overline{H})| - 1)}{2}}{|V(\overline{H})| - 1} = |V(\overline{H})|$$

If $|S| = |V(\overline{H})|$ then both inequalities in the above equation must hold as equalities, so

$$|E(\overline{H})| = \frac{|V(\overline{H})|(|V(\overline{H})| - 1)}{2}$$

which implies that $\overline{H}$ is a clique. ■

Note that 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, say, $|C| > \alpha > \frac{|V|}{2}$ 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 *Suppose G is not highly connected but has diameter 2. Let H and $\overline{H}$ be the induced subgraphs obtained by removing a minimum cut S from G, where $|V(\overline{H})| \leq |V(H)|$. Then (1) every vertex in $\overline{H}$ is incident on S, (2) $\overline{H}$ is a clique, and (3) if $|V(\overline{H})| > 1$ then every vertex in $\overline{H}$ is incident on a single edge of S.*

Proof: Case 1, $|S| = \frac{|V|}{2}$: If $|V(\overline{H})| = 1$, the theorem is trivially true. Suppose $|V(\overline{H})| > 1$. By Lemma 2, H and $\overline{H}$ are cliques on $\frac{|V|}{2}$ vertices. If there exists $x \in \overline{H}$ so that x is not incident on S then $deg(x) < \frac{|V|}{2}$ so, the partition $(\{x\}, V\backslash\{x\})$ has a smaller cut value than $|S|$, a contradiction.

Case 2, $|S| < \frac{|V|}{2}$: We first show that $|V(\overline{H})| < |V(H)|$: Suppose $|V(H)| = |V(\overline{H})| = \frac{|V|}{2}$. Then since $|S| < |V(H)|$ and $|S| < |V(\overline{H})|$, there exist $u \in H$ and $v \in \overline{H}$ such that u is not adjacent to any vertex in $\overline{H}$, and v is not adjacent to any vertex in H. But then $d(u, v) > 2$, a contradiction to the fact that $diam(G) \leq 2$.
Since $|V(H)| > \frac{|V|}{2} > |S|$, there exist a vertex $v \in H$ that is not incident on S. If there exists a vertex $u \in \overline{H}$ that is not incident on S then again we get $d(u, v) > 2$, a contradiction. This proves assertion (1).
By assertion (1), $|S| \geq |V(\overline{H})|$. If $|V(\overline{H})| = 1$, $\overline{H}$ is trivially a clique. Suppose $|V(\overline{H})| > 1$. By Lemma 2, $|S| \leq |V(\overline{H})|$. Therefore, $|S| = |V(\overline{H})|$. The second part of Lemma 2 now implies that $\overline{H}$ is a clique. ■

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

Proof: (1) Let $G = (V, E)$ be a highly connected subgraph with n vertices. We saw in the proof of Theorem 1 that $k(G) \leq \delta(G)$. By the definition of a HCS $\frac{n}{2} < k(G)$. Therefore, $\frac{n}{2} < \delta(G)$, and a lower bound for the total number of edges is $|E| > \frac{n}{2} \times n \times \frac{1}{2} = \frac{n^2}{4}$.

(2) If an iteration of the HCS algorithm splits an n-vertex graph into two components, then the number of edges between these components is at most $\frac{n}{2}$. ■

By Theorem 1 each cluster produced by the HCS algorithm has diameter at most two. This is a strong indication to the homogeneity, as the only better possibility in terms of the diameter is that *every* two vertices of a cluster are connected by an edge. This condition is too stringent since it does not allow false negative errors in determining similarities. Moreover, its use requires solving the NP-hard maximum clique problem. By Theorem 4(a) we see that each cluster is at least half as dense as a clique, which is another strong indication of homogeneity.

The above theorems also give a strong indication of the separation property of the solution provided by the HCS algorithm, in that any non-trivial set split by the algorithm is unlikely to have diameter two: Suppose the algorithm splits the subgraph G' into non-trivial sets C_1 and C_2, and $diam(G') \leq 2$. Let S be the set of minimum cut edges causing that split. W.l.o.g. suppose $t = |C_1| \leq |C_2|$. By Theorem 3 every vertex of C_1 is incident on S. How likely is it that G' is a true cluster or a part thereof? Each vertex in C_1 is adjacent only to a *single* vertex in C_2, and is not adjacent to the rest of the vertices in C_2. As $|C_2| \geq |t|$, it follows that in the true cluster containing $C_1 \cup C_2$, only t out of the t^2 or more edges between C_1 and C_2 are present, and they manifest a highly structured pattern. In contrast, within C_1, all $\binom{t}{2}$ edges are present, again by Theorem 3. Therefore, unless t is very small we have a situation that is highly unlikely to be caused by random noise within a cluster. Hence, with the exception of this unlikely situation, any non trivial set split by the algorithm has diameter at least three.

Another indication of separation is given in Theorem 4: The number of edges removed by each iteration of the HCS algorithm is at most linear in the size of the underlying subgraph, compared to a quadratic number of edges within final clusters. This indicates separation, unless the sizes are very small. Note, however, that this does not imply that the number of edges between any two clusters that are eventually produced by the algorithm is at most linear.

4 Practice

We describe below several heuristic improvements that speed up the algorithm and improve its performance in practice.

4.1 Iterated HCS: When there are several minimum cuts in a graph, the HCS algorithm might choose a minimum cut which is not best from a clustering point of view. In many cases this process will break clusters into singletons. (For example, a different choice of minimum cuts by the algorithm for the graph in Figure 2 may split x from G_2 and eventually find the clusters G_1 and G_3, leaving x, y, z as singletons.) A possible solution is to perform several iterations of the HCS algorithm until no new cluster is found. The iterated HCS adds theoretically another $O(n)$ factor to the running time, but in practice only very few (1-5) iterations are usually needed.

4.2 Singletons adoption: Elements left as singletons by the initial clustering process can be "adopted" by clusters based on similarity to the cluster: For each singleton element x we compute the number of neighbors it has in each cluster and in the singletons set $\mathcal{S}$. If the maximum number of neighbors is sufficiently large, and is obtained by one of the clusters (rather than by $\mathcal{S}$), then x is added to that cluster. The process is repeated up to a prescribed number of times in order to accommodate changes in clusters as a result of previous adoptions.

4.3 Removing Low Degree Vertices: When the input graph contains vertices with low degrees, one iteration of the mincut 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. Removing low degree vertices from G eliminates such iterations, and significantly reduces the running time. A refinement of the algorithm that incorporates this idea as well as the singleton adoption and the iterated HCS is shown in Figure 3. $d_1, d_2, \ldots d_p$ is a decreasing sequence of integers given as external input to the algorithm.

```
HCS_LOOP(G(V,E))
begin
   for (i=1 to p) do
      remove clustered vertices from G
      H ← G
      repeatedly remove all vertices of degree < d_i from H
      until (no new cluster is found by the HCS call) do
            HCS(H)
            perform singletons adoption
            remove clustered vertices from H
      end until
   end for
end
```

Figure 3: The improved HCS algorithm.

The algorithm, together with the heuristics, was implemented and tested on both simulated and real data, with very good results. Detailed description of the experiments can be found in [8, 9]. For completeness, we summarize these results very briefly. We used the following score to evaluate the quality of clustering: A clustering solution for a set of n elements can be represented by an $n \times n$ matrix M where $M_{ij} = 1$ iff i and j are in the same cluster according to the solution and $M_{ij} = 0$ otherwise. If T denotes the matrix of the true solution, then the *Minkowski score* of M is $||T - M||/||T||$ [21, 10]. Hence, a perfect solution will obtain a score of zero, and the smaller the score the better the solution.

The algorithm was first applied to simulated of gene expression data. An extant greedy algorithm used in practice on such data [17] was also applied for comparison. On ten different groups of datasets, varying in sizes from 60 to 980 elements with 3-13 clusters and high noise rate, HCS achieved an average score below 0.2. In comparison, the greedy algorithm deteriorated from an average score of 0.4 for the smallest problems to 1.4 for the largest. Stability of the solution quality, as measured by the variance of the score, was also much smaller in our algorithm. HCS also manifested robustness with respect to higher noise levels.

Next, the algorithms were applied in a blind test to real gene expression data provided by H. Lehrach's group of Max Planck Institute, Berlin. The data consisted of 2329 elements partitioned into 18 clusters varying in size from 1 to 708 elements. HCS identified 16 clusters and left 206 as singletons, reaching a score of 0.71. Greedy generated 66 clusters and left 478 singletons, and got a score of 0.77. Hence HCS was superior in all parameters. Running time of HCS on a 194 MHZ SGI machine was 43 minutes.

5 Concluding Remarks

We have presented a clustering algorithm based on high connectivity in graphs, and demonstrated that it generates solutions with desirable properties for clustering. The algorithm has low polynomial complexity. It is also efficient in practice: Our initial implementation, after some heuristic improvements as described in Section 4, handles well problems with up to thousands of elements in a reasonable computing time [7].

As noted in the introduction, graph connectivity has been previously used for clustering. Our novel definition of highly connected subgraphs gives a stopping criterion, by defining clusters as subgraphs with connectivity that is above half the number of vertices. This has several advantages: It gives clusters with provable good properties (Section 3), and it precludes the need to know in advance (or guess) the number of clusters as in [22].

The HCS algorithm generates clusters with diameter two. This is a strong indication of homogeneity, as any two vertices are either adjacent or share one or more common neighbors. This property is not satisfied by the solutions in [14, 22]. (The 3-connected component in [14, Fig. 1] has diameter 3. Moreover, a path of arbitrary length is a cluster according to [14]).

In contrast with [22], the HCS algorithm computes minimum cuts in the smaller subgraphs which were obtained by removing the mincut edges of previous partitions. It seems that the HCS way of computing mincuts is more suitable for clustering, because the mincut edges of previous partitions correspond to erroneous edges which connect mistakenly entities from different clusters, so there is no reason to take them into account again in subsequent partitions. For example, on the graph in Figure 4, given the number of clusters 3 as input, the algorithm in [22] will find an isolated vertex $v \in \{a, b, c, d\}$, $G_1 \setminus \{v\}$, and $(G_2 \cup G_3) \setminus \{v\}$. In contrast, HCS finds the three more plausible clusters G_1, G_2, G_3. On the graph in Figure 2, [22] finds G_1, $\{x\}$, and $G_3 \cup \{y, z\}$.

We have chosen not to view as an HCS (and thus a cluster) an n-vertex subgraph H with connectivity exactly $\frac{n}{2}$, even though its diameter is 2. By Theorem 3, in that case H consists of two $\frac{n}{2}$-vertex cliques connected by a perfect matching. Unless n is very small, this is more likely to be a union of two clusters. (Subgraphs with $n = 2$ may be handled as a special case in implementations.) In contrast, if a minimum cut exceeds $\frac{n}{2}$, it must separate only a single vertex from the graph.

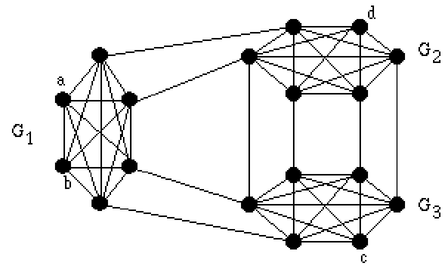

Figure 4: A similarity graph of three clusters G_1, G_2, G_3, with some false positive edges. Here the basic HCS clustering performs better than Wu and Leahy's clustering.

Possible future improvements include finding maximal highly connected subgraphs (e.g. using Matula's cohesiveness function [14]), and finding a weighted minimum cut in an edge-weighted graph. Most of our theoretical results carry over to weighted graphs. Further theoretical questions on the properties of the algorithm are also of interest: Can we determine the optimal value of the threshold on the similarity values that is used to form the similarity graph? Can we determine a probabilistic threshold for the level of noise under which good (or perfect) clustering is guaranteed, with high probability? Proving additional (deterministic or probabilistic) properties of HCS clusters is also of interest.

References

[1] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, 1973.

[2] F. Buckley and F. Harary. *Distance in Graphs.* Addison-Wesley, 1990.

[3] S. Even. *Graph Algorithms.* Computer Science Press, Potomac, Maryland, 1979.

[4] R.E. Gomory and T.C. Hu. Multy-terminal networks flows. *SIAM J. Apll. Math*, 9:551–570, 1961.

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

[6] J.A. Hartigan. *Clustering Algorithms.* John Wiley and Sons, 1975.

[7] 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.

[8] 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, April 1999.

[9] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, and R. Shamir. An algorithm for clustering cDNA fingerprints. *Genomics*, 66(3):249–256, 2000. A preliminary version appeared in Proc. RECOMB '99, pp. 188–197.

[10] N. Jardine and R. Sibson. *Mathematical Taxonomy.* John Wiley and Sons, London, 1971.

[11] D. Karger. Minimum cuts in near-linear time. In *Proc. STOC 96.* ACM Press, 1996.

[12] D.W. Matula. The cohesive strength of graphs. In G. Chartrand and S.F. Kapoor, editors, *The Many Facets of Graph Theory*, pages 215–221, Berlin, 1969. Springer-Verlag. Lecture Notes in Mathematics No.110.

[13] 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.

[14] D.W. Matula. k-Components, clusters and slicings in graphs. *SIAM J. Appl. Math.*, 22(3):459–480, 1972.

[15] D.W. Matula. Graph theoretic techniques for cluster analysis algorithms. In J. Van Ryzin, editor, *Classification and Clustering*, pages 95–129. Academic Press, 1977.

[16] D.W. Matula. Determining edge connectivity in $O(nm)$. In *Proceedings 28th IEEE Symposium on Foundations of Computer Science*, pages 249–251, 1987.

[17] A. Milosavljevic, Z. Strezoska, M. Zeremski, D. Grujic, T. Paunesku, and R. Crkvenjakov. Clone clustering by hybridization. *Genomics*, 27:83–89, 1995.

[18] B. Mirkin. *Mathematical Classification and Clustering.* Kluwer, 1996.

[19] H. Nagamochi and T. Ibaraki. Computing edge connectivity in multigraphs and capacitated graphs. *SIAM J. Disc. Math.*, 5:54–66, 1992.

[20] P.H.A. Sneath and R.R. Sokal. *Numerical Taxonomy.* Freeman, San Francisco, 1973.

[21] R. R. Sokal. Clustering and classification: Background and current directions. In J. Van Ryzin, editor, *Classification and Clustering*, pages 1–15. Academic Press, 1977.

[22] 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.