A NOVEL CLUSTERING METHOD FOR ANALYSIS OF BIOLOGICAL NETWORKS USING MAXIMAL COMPONENTS OF GRAPHS

MORIHIRO HAYASHIDA AND TATSUYA AKUTSU

Bioinformatics Center, Institute for Chemical Research, Kyoto University
Uji-city, Kyoto 611-0011, Japan
E-mail: {morihiro, takutsu}@kuicr.kyoto-u.ac.jp

HIROSHI NAGAMOCHI

Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University
Kyoto-city, Kyoto 606-8501, Japan
E-mail: nag@amp.i.kyoto-u.ac.jp

This paper proposes a novel clustering method for analyzing biological networks. In this method, each biological network is treated as an undirected graph and edges are weighted based on similarities of nodes. Then, maximal components, which are defined based on edge connectivity, are computed and the nodes are partitioned into clusters by selecting disjoint maximal components. The proposed method was applied to clustering of protein sequences and was compared with conventional clustering methods. The obtained clusters were evaluated using *P*-values for GO (GeneOntology) terms. The average *P*-values for the proposed method were better than those for other methods.

1. Introduction

Clustering is one of fundamental techniques in bioinformatics. Indeed, many clustering methods have been developed and/or applied for analyzing various kinds of biological data. Among them, such hierarchical clustering methods as the single-linkage, complete-linkage and average-linkage methods have been widely used [3,9]. However, these clustering methods are based on similarities between two elements or two clusters, and relations with other elements or clusters are not so much taken into account.

Relations between biological entities are often represented as networks or (almost equivalently) graphs. For example, nodes are proteins in a protein-protein interaction network, and two nodes are connected by an edge if the corresponding proteins interact with each other. For another example, nodes are again proteins in a sequence similarity network of proteins, and two nodes are connected by an edge if the corresponding protein sequences are similar to each other. Moreover, in this case, similarity scores are assigned as weights of edges. Since these networks are considered to have much information, clustering based on network structures might be useful. Of course, conventional clustering methods can be applied to clustering of nodes in these networks [3,9]. But, information on network structure is not so much taken into account by these methods. For an extreme example, suppose that the network is a complete graph and all edges have the same weight. Then, all the

nodes should be put into one cluster and sub-clusters should not be created. However, conventional clustering methods create many sub-clusters. Therefore, clustering methods that utilize structural information on a network should be developed. Though clustering methods utilizing structural information have been developed [7,12], many of these are heuristic and/or recursive and thus it is unclear which properties are satisfied for the final clusters.

On the other hand, in graph theory and graph algorithms, the Gomory-Hu tree is well-known [5] where it is defined for an undirected network with weighted edges. This tree essentially contains all information on minimum cuts for all pairs of nodes. It is known that a Gomory-Hu tree can be computed efficiently using a maximum flow algorithm. Furthermore, maximal components can be efficiently computed from a Gomory-Hu tree [11], where a maximal component is a set of nodes with high connectivity (the precise definition is given in Sec. 2). It is known that a set of maximal components constitutes a laminar structure, which is essentially a hierarchical structure.

Based on the above facts, we develop a novel clustering method for an undirected network. In this method, nodes are partitioned into clusters by selecting disjoint maximal components. The method works in $O(n^2 m \log(n^2/m))$ time, where n and m are the numbers of nodes and edges, respectively. The Gomory-Hu tree was already applied to analysis of protein folding pathways [8,10,13]. However, to our knowledge, it was not applied to analysis of large scale protein sequence networks. Moreover, as to be shown in Sec. 3, our method employs additional ideas to effectively utilize the Gomory-Hu tree.

In this paper, we apply the proposed method to clustering of protein sequences and compare with the single-linkage, complete-linkage and average-linkage methods. We evaluate the computed clusters using *P*-values for GO (GeneOntology) terms. The results suggest the effectiveness of the proposed method.

The organization of the paper is as follows. In Sec. 2, we briefly review maximal components of undirected graphs and conventional clustering methods. In Sec. 3, we present our clustering method based on maximal components. In Sec. 4, we show the results on computational experiment. Finally, we conclude with future work.

2. Preliminaries

In this section, we review edge-connectivity and maximal components [11]. We also review three conventional hierarchical clustering methods: single-linkage, average-linkage and complete-linkage clustering methods.

2.1. *Edge-connectivity*

Let $G = (V, E)$ be an undirected edge-weighted graph with a vertex set V and an edge set E, where each edge e is weighted by a nonnegative real $c_G(e) \in \Re^+$. We define the edge-connectivity $\lambda_G(u, v)$ between two nodes u and v as follows:

$$\lambda_G(u, v) = \min_{\{X \subseteq V | u \in X, v \in V - X\}} \sum_{p \in X, q \in V - X} c_G(p, q). \quad (1)$$

A subset X of V is called (u, v)-cut if $u \in X$ and $v \in V - X$, or $u \in V - X$ and

$v \in X$. Among them, a (u, v)-cut X which gives a minimum $\lambda_G(u, v)$ is called a minimum (u, v)-cut.

2.2. *Maximal Components*

Definition 1. A subset X of V is called a *maximal component* if it satisfies the following conditions,

$$\forall u, v \in X \qquad \lambda_G(u, v) \geq l, \tag{2}$$

$$\forall u \in X, \forall v \in V - X \ \lambda_G(u, v) < l, \tag{3}$$

where $l = \min_{u,v \in X} \lambda_G(u, v)$. Such a subset X is also called an *l-edge-connected component*.

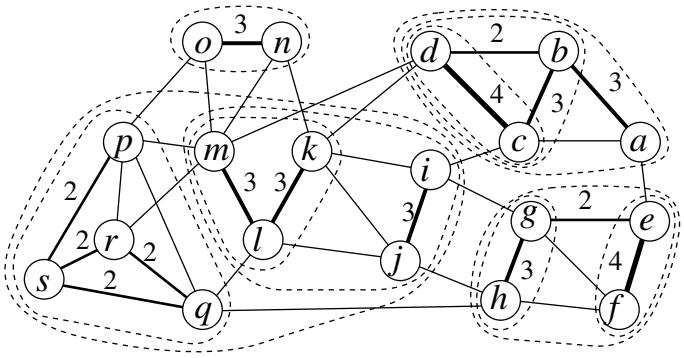

Figure 1. Illustration for maximal components of a graph $G = (V, E)$ with $V = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s\}$ and E, where each number denotes the weight of the edge, and edges without numbers are weighted by 1. Each set of nodes surrounded with a dashed line is a maximal component of G. For example, the set $X = \{a, b, c, d\}$ is a maximal component because $\lambda_G(u, v) \geq 5$ for any $u, v \in X$, $\lambda_G(u, v) < 5$ for any $u \in X$ and $v \in V - X$, and $\min_{u,v \in X} \lambda_G(u, v) = 5$. It is also called a 5-edge-connected component.

Figure 1 shows an example of maximal components. Definition 1 means that the internal nodes of a maximal component are connected with each other more strongly than with any other external nodes. Moreover, nodes of internal maximal components are connected with each other more strongly than (and equally to) those of external maximal components which include the internal maximal components.

Definition 2. A family $\boldsymbol{\chi} \subseteq 2^V$ is called *laminar* if $X \cap Y = \emptyset$, $X \subseteq Y$, or $Y \subset X$ for any sets $X, Y \in \boldsymbol{\chi}$.

A laminar family $\boldsymbol{\chi}$ is represented by a rooted tree $\boldsymbol{\tau} = (\boldsymbol{\nu}, \boldsymbol{\epsilon})$. The node set $\boldsymbol{\nu}$ is defined by $\boldsymbol{\nu} = \boldsymbol{\chi} \cup \{V\}$, where V corresponds to the root of $\boldsymbol{\tau}$. Let t_X denote a node corresponding to a set $X \in \boldsymbol{\nu}$. For two nodes t_X and t_Y in $\boldsymbol{\tau}$, t_X is a child of t_Y if and only if $X \subset Y$ holds and $\boldsymbol{\chi}$ contains no set Z with $X \subset Z \subset Y$.

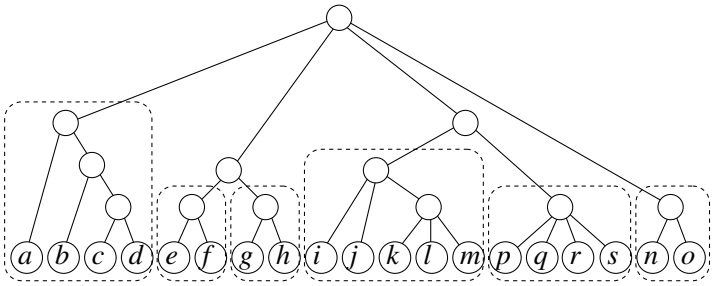

Figure 2. The rooted tree representation of maximal components $\boldsymbol{\chi}(G)$ of the graph G in Fig. 1. The six sets of nodes surrounded by dashed lines are the resulting clusters provided by the procedure SelectLaminar.

Theorem 1. *Let $\boldsymbol{\chi}(G)$ denote the set of all maximal components of G. Then, $\boldsymbol{\chi}(G)$ is a laminar family.*

Proof. We assume that there exist three nodes x, y and z so that $x \in X - Y$, $y \in Y - X$, and $z \in X \cap Y$ for two maximal components $X, Y \in \boldsymbol{\chi}(G)$, where X is an l-edge-connected component and Y is an h-edge connected component. We can assume without loss of generality that $l \geq h$. From $x, z \in X$ and Eq. 2 of the definition of maximal components for X, we have $\lambda_G(x, z) \geq l \geq h$. On the other hand, from $x \notin Y, z \in Y$ and Inequality (3) for Y, we have $\lambda_G(x, z) < h$. It contradicts our assumption. □

2.3. *Linkage methods*

We briefly review three linkage clustering methods: single linkage (or nearest neighbor method), complete linkage (or farthest neighbor method), and average linkage. Each method starts with a set of clusters, where each cluster consists of a single distinct node. Then, two clusters having the minimum distance are merged into one cluster. This procedure is repeated until there is only one cluster. The distance $D(X, Y)$ between two clusters X and Y is defined in a different way depending on a clustering method:

$$D(X,Y) = \begin{cases} \min\limits_{x \in X, y \in Y} d(x,y) & \text{(for single linkage)} \\ \max\limits_{x \in X, y \in Y} d(x,y) & \text{(for complete linkage)} \\ \dfrac{1}{|X||Y|} \sum\limits_{x \in X, y \in Y} d(x,y) & \text{(for average linkage)} \end{cases}, \tag{4}$$

where $d(x, y)$ denotes the distance between two nodes x and y.

It should be noted that the distance between two nodes should be small if the similarity between these nodes is high, whereas the weight of the edge between these two nodes should be large. Since we are going to use the similarity score (which is high for similar nodes), we use modified versions of these clustering algorithms. In the modified versions, the clusters with the maximum score are merged, instead of the clusters with the minimum distance. Moreover, '$\min$' and '$\max$' in Eq. 4 are exchanged.

3. Selection of Disjoint Clusters from Hierarchical Structure

The set of all maximal components $\boldsymbol{\chi}(G)$ of a graph G provides a hierarchical structure which can be represented as a rooted tree $\boldsymbol{\tau}(G)$ because the set $\boldsymbol{\chi}(G)$ is a laminar family. This structure gives a kind of hierarchical clustering. However, what we need is a set of disjoint clusters because we are interested in classification of protein sequences. That is, input nodes should be partitioned into disjoint clusters. Thus, we propose a method to find disjoint clusters from $\boldsymbol{\chi}(G)$. In our method, a set of maximal components $\boldsymbol{\chi}(G)$ of the graph G is first computed using a Gomory-Hu tree. And then, disjoint maximal components are selected in a bottom-up manner, based on the tree structure $\boldsymbol{\tau}(G)$. The detailed procedure is given below.

Procedure SelectLaminar
  Input : a laminar family $\boldsymbol{\chi}$
  Output : a set of clusters $\boldsymbol{\chi}_c \subseteq \boldsymbol{\chi}$
Begin
  $\boldsymbol{\tau} :=$ the rooted tree made from $\boldsymbol{\chi}$
  $\boldsymbol{\chi}_c := \emptyset$
  repeat
    $X_p :=$ a parent node of not marked deepest leaves of $\boldsymbol{\tau}$
    repeat
      $X_s := X_p$
      $X_p :=$ the parent node of X_s
    until X_p has a child X_t except X_s such that $|X_t| \geq 2$
    Add all the child nodes of X_p to $\boldsymbol{\chi}_c$
    Mark all the descendant leaves of X_p in $\boldsymbol{\tau}$
  until all the leaves of $\boldsymbol{\tau}$ are marked
  return $\boldsymbol{\chi}_c$
End

It should be noted that $|X_t|$ denotes the number of nodes in G that are contained in X_t. This procedure outputs a subset $\boldsymbol{\chi}_c = \{X_1, \ldots, X_m\}$ from the laminar family $\boldsymbol{\chi}(G)$ of all maximal components of a graph G such that $X_i \cap X_j = \emptyset$ holds for any two sets $X_i \neq X_j \in \boldsymbol{\chi}_c$, and $\bigcup_{i=1}^{m} X_i = V$ holds. Figure 2 shows an example. This procedure provides the clusters according to the hierarchical structure.

4. Experimental Results

4.1. *Data and Implementation*

In order to evaluate the proposed clustering method, we applied clustering methods to classification of protein sequences based on the pairwise similarity. We used 5888 protein sequences (The file name is "orf_trans.20040827.fasta") from Saccharomyces Genome Database (SGD) [6]. This file contains the translations of all systematically named ORFs except dubious ORFs and pseudo-genes. We calculated the similarities between all pairs of

the proteins using a BLAST search [1] with an *E*-value threshold of 0.1. An edge between two nodes exists only when the *E*-value between the proteins is less than or equal to 0.1. All isolated nodes (i.e., nodes with degree 0) are removed. As a result, 32484 pairwise similarities and 4533 nodes were detected.

As an edge-weight, we used the integer part of $-3000 \log_{10} h$ for the *E*-value h of $10^{-\frac{1000}{3}} < h \leq 0.1$, and 10^6 for $0 \leq h \leq 10^{-\frac{1000}{3}}$. This mapping was injective for all the *E*-values of the data. It should be noted that the similarity between proteins is large when the *E*-value is small, and comparison operations of floating point numbers can cause incorrect results.

We solved maximum flow problems with HIPR (version 3.5) [15] which is an implementation of the algorithm developed by Goldberg and Tarjan [4], and constructed a Gomory-Hu tree [5] for an edge-weighted graph G to obtain all the maximal components of G from the tree.

4.2. *Results*

To evaluate the performance of our clustering method, we used GO-TermFinder (version 0.7) [2]. To find the most suitable GO term for a specified list (cluster) of genes, this software calculates a *P*-value using the hypergeometric distribution as follows:

$$P = 1 - \sum_{i=0}^{k-1} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}} = \sum_{i=k}^{n} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}} \tag{5}$$

where N is the total number of genes, M is the total number of genes annotated by the specific GO term, n is the number of genes in the cluster, and k is the number of genes annotated by the specific GO term in the cluster. *P*-value means the probability of seeing k or more genes with an annotation by a GO term among n genes in the list, given that M in the population of N have that annotation. For example, $P = 1$ holds if none of the genes in the specified list are annotated by the GO term. On the other hand, if all the genes are annotated, $P = \frac{M(M-1)\cdots(M-n+1)}{N(N-1)\cdots(N-n+1)}$ is very small because M is usually much smaller than N.

In order to avoid that a lot of false positive GO terms are chosen, GO-TermFinder can use corrected *P*-values. We employed these corrected *P*-values to evaluate clustering results.

We used three types of ontologies on biological processes, cellular components, and molecular functions (Their file names are "process.ontology.2005-08-01","component.ontology.2005-08-01", and "function.ontology.2005-08-01"). We obtained these files also from SGD [6].

We compared the proposed method with other clustering methods using single linkage, complete linkage, and average linkage. These clustering methods usually produce a hierarchical clustering. In order to obtain non-hierarchical clustering results, we applied our proposed procedure in the previous section, SelectLaminar, to their results.

Table 1 shows the averages of logarithms of corrected *P*-values over all 4533 proteins. Among these proteins, there were some proteins which could not be annotated by GO-TermFinder. Therefore, we regarded a corrected *P*-value as 1 for such proteins, and calculated the averages. We see from the table that our clustering method using maximal components outperformed other methods. For every ontology, the average of our method was lower than that of others. It means that our method classified protein sequences into protein functions better than others.

Table 1. Results for three ontologies on biological processes, cellular components, and molecular functions, by four clustering methods using maximal components, single linkage, complete linkage, and average linkage. Left column: the average of logarithm of corrected *P*-values. Right: the number of annotated proteins.

Method	Process		Component		Function	
Maximal component	-8.9462	2618	-5.9189	2641	-10.657	2624
Single linkage	-5.2346	2947	-4.5076	2970	-4.7721	2903
Complete linkage	-3.0674	3258	-2.3149	3391	-3.8539	3050
Average linkage	-3.2556	3692	-2.4423	3761	-4.1007	3508

Figures 3, 4 and 5 show logarithms of corrected *P*-values on 800 lowest proteins for the ontologies on biological processes, cellular components, and molecular functions, respectively. For every ontology, corrected *P*-values of our method were lower than others. The distributions of complete linkage and average linkage had similar behavior. For the ontologies on biological processes and cellular components, corrected *P*-values of single linkage were close to those of our method. In particular, our method provided good results for molecular functions.

Table 2 shows GO terms with lowest 8 corrected *P*-value for the ontology of biological processes in resulting clusters of clustering methods using maximal components, single linkage, complete linkage, and average linkage. In both complete linkage and average linkage, the same GO term (GO:0006319 Ty element transposition) was annotated to the first and second lowest clusters. It means that a cluster having the GO term was divided into two or more clusters by the methods.

As for CPU time, the proposed method is reasonably fast. Though the worst case time complexity of the proposed method is $O(n^2 m \log(n^2/m))$, it is expected to work faster in practice. Indeed, the proposed method took 6.3 sec. for clustering of a graph with 4533 nodes on a Linux PC with Xeon 3.6GHz CPU and 4GB memory. Though the single-linkage clustering took only 0.024 sec., our proposed method produced better results.

5. Conclusion

We developed a clustering method using maximal components, where a maximal component can be characterized as a subgraph having maximal edge connectivity. We compared

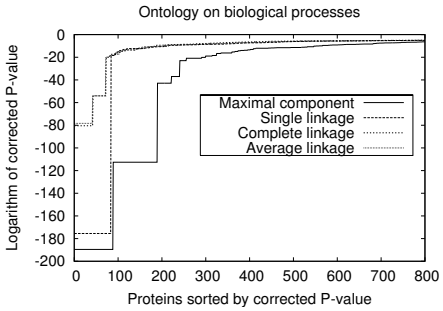

Figure 3. Logarithms of corrected *P*-values on 800 lowest proteins for ontology on biological processes.

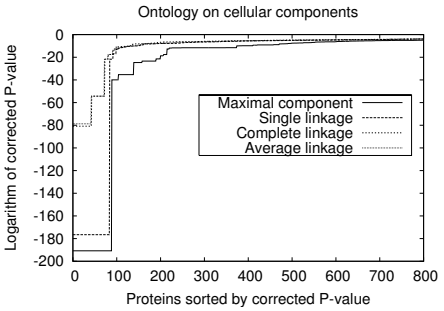

Figure 4. Logarithms of corrected *P*-values on 800 lowest proteins for ontology on cellular components.

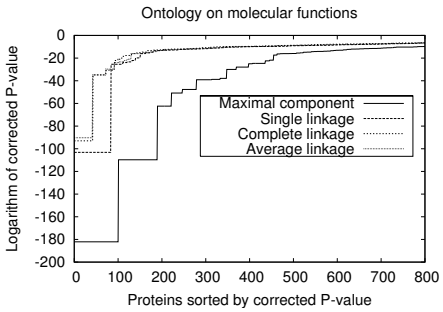

Figure 5. Logarithms of corrected *P*-values on 800 lowest proteins for ontology on molecular functions.

the proposed method with the single linkage, complete linkage, and average linkage clustering methods using protein sequence data. Our proposed method outperformed these three methods in terms of the corrected *P*-values provided by GO-TermFinder, and classified protein sequences into protein functions better than the three methods.

Although we did not compare clustering methods other than the linkage methods with our method in this study, many clustering methods have been proposed. For example, the k-means method [14] is well known as a non-hierarchical clustering method. However, it cannot be directly applied to edge-weighted graphs because it is difficult to define the center of a cluster and the distance between the center and any node in the graph.

There are several future works. We used $\log$ of *E*-values as edge-weights. However, this weighting method is not necessarily the best. Thus, finding better weighting method is important future work. We developed a simple method in order to select disjoint clusters from a set of maximal components. However, better results may be obtained by using a more elaborated method. Thus, improvement of selection of disjoint clusters should be done. We have applied the proposed clustering method to clustering of protein sequences. However, our method is not limited to analysis of protein sequences. For example, clustering of gene expression data is one of extensively studied problems. Therefore, application to analysis of gene expression data is also important future work.

Acknowledgments

This work is supported in part by Grants-in-Aid #1630092 and "Systems Genomics" from the Ministry of Education, Science, Sports, and Culture of Japan.

References

1. S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller and D.J. Lipman, Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Res.*, 25:3389-3402, 1997.
2. E.I. Boyle, S. Weng, J. Gllub, H. Jin, D. Botstein, J.M. Cherry and G. Sherlock, GO::TermFinder–open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes, *Bioinformatics*, 20:3710-3715, 2004.
3. A.J. Enright and C.A. Ouzounis, GeneRAGE: a robust algorithm for sequence clustering and domain detection, *Bioinformatics*, 16:451–457, 2004.
4. A. Goldberg and R. Tarjan, A new approach to the maximum flow problem, *Journal of the Association for Computing Machinery*, 35:921-940, 1988.
5. R.E. Gomory and T.C. Hu, Multi-terminal network flows, *SIAM Journal of Applied Mathematics*, 9:551-570, 1961.
6. E.L. Hong, R. Balakrishnan, K.R. Christie, M.C. Costanzo, S.S. Dwight, S.R. Engel, D.G. Fisk, J.E. Hirschman, M.S. Livestone, R. Nash, J. Park, R. Oughtred, M. Skrzypek, B. Starr, C.L. Theesfeld, R. Andrada, G. Binkley, Q. Dong, C. Lane, B. Hitz, S. Miyasato, M. Schroeder, A. Sethuraman, S. Weng, K. Dolinski, D. Botstein and J.M. Cherry, Saccharomyces Genome Database, ftp://ftp.yeastgenome.org/yeast/
7. H. Kawaji, Y. Takenaga and H. Matsuda, Graph-based clustering for finding distant relationships in a large set of proteins, *Bioinformatics*, 20:243–252, 2004.
8. J.M. Kleinberg, Efficient algorithms for protein sequence design and the analysis of certain evolutionary fitness landscapes, *Proc. 3rd Int. Conf. Computational Molecular Biology (RECOMB)*, 226–237, 1999.
9. E.V. Koonin, R.L. Tatusov and K.E. Rudd, Sequence similarity analysis of Escherichia coli proteins: Functional and evolutionary implications, *Proc. Natl. Acad. Sci. USA*, 92:11921–11925, 1995.
10. S.V. Krivov, and M. Karplus, Hidden complexity of free energy surfaces for peptide (protein) folding, *Proc. Natl. Acad. Sci. USA*, 101:14766–14770, 2004.
11. H. Nagamochi, Graph algorithms for network connectivity problems, *Journal of the Operating Research Society of Japan*, 47:199–223, 2004.
12. G. Yona, N. Linial and M. Linial, ProtoMap: automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space, *PROTEINS: Structure, Function, and Genetics*, 37:360–378, 1999.
13. M.J. Zaki, V. Nadimpally, D. Bardhan and C. Bystroff, Predicting protein folding pathways, *Bioinformatics*, 20:i386–i393, 2004.
14. J.B. MacQueen, Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.
15. http://www.avglab.com/andrew/soft.html

Table 2. GO terms with lowest 8 corrected *P*-value for the ontology of biological processes in resulting clusters of clustering methods using maximal components, single linkage, complete linkage, and average linkage.

Rank	Maximal component	
1	GO:0006319 (Ty element transposition)	2.7522e-190
2	GO:0006468 (protein amino acid phosphorylation)	2.4181e-113
3	GO:0008643 (carbohydrate transport)	1.2509e-43
4	GO:0006865 (amino acid transport)	1.0052e-37
5	GO:0006511 (ubiquitin-dependent protein catabolism)	9.4800e-24
6	GO:0006810 (transport)	1.2224e-21
7	GO:0006081 (aldehyde metabolism)	8.1134e-21
8	GO:0016567 (protein ubiquitination)	1.1405e-19
Rank	Single linkage	
1	GO:0006319 (Ty element transposition)	3.0396e-176
2	GO:0006081 (aldehyde metabolism)	4.0950e-19
3	GO:0006530 (asparagine catabolism)	3.5363e-16
4	GO:0006166 (purine ribonucleoside salvage)	5.0151e-15
5	GO:0045039 (mitochondrial inner membrane protein import)	3.1055e-14
6	GO:0046839 (phospholipid dephosphorylation)	7.8293e-14
7	GO:0005992 (trehalose biosynthesis)	3.4570e-13
8	GO:0006913 (nucleocytoplasmic transport)	4.4109e-13
Rank	Complete linkage	
1	GO:0006319 (Ty element transposition)	3.7058e-81
2	GO:0006319 (Ty element transposition)	9.1783e-55
3	GO:0008645 (hexose transport)	1.1156e-20
4	GO:0006319 (Ty element transposition)	4.1098e-18
5	GO:0000209 (protein polyubiquitination)	5.7229e-17
6	GO:0006530 (asparagine catabolism)	3.5363e-16
7	GO:0006081 (aldehyde metabolism)	2.1634e-15
8	GO:0006166 (purine ribonucleoside salvage)	5.0151e-15
Rank	Average linkage	
1	GO:0006319 (Ty element transposition)	4.4023e-79
2	GO:0006319 (Ty element transposition)	9.1783e-55
3	GO:0008645 (hexose transport)	1.1156e-20
4	GO:0006081 (aldehyde metabolism)	4.0950e-19
5	GO:0006319 (Ty element transposition)	4.1098e-18
6	GO:0006530 (asparagine catabolism)	3.5363e-16
7	GO:0006166 (purine ribonucleoside salvage)	5.0151e-15
8	GO:0045039 (mitochondrial inner membrane protein import)	3.1055e-14