Vol. 47 No. SIG 17(TBIO 1) IPSJ Transactions on Bioinformatics Nov. 2006

Original Paper

Comparison of Protein Complexes Predicted from PPI Networks by DPClus and Newman Clustering Algorithms

Hisashi Tuji,[†] Md. Altaf-Ul-Amin,[†] Masanori Arita,[††]
Hirokazu Nishio,[††] Yoko Shinbo,[†] Ken Kurokawa[†]
and Shigehiko Kanaya[†]

A Protein-Protein Interaction network, what we call a PPI network is considered as an important source of information for prediction of protein functions. However, it is quite difficult to analyze such networks for their complexity. We expected that if we could develop a good visualizing method for PPI networks, we could predict protein functions visually because of the close relation between protein functions and protein interactions. Previously, we proposed one, which is based on clustering concepts, by extracting clusters defined as relatively densely connected group of nodes. But the results of visualization of a network differ very much depending on the clustering algorithm. Therefore, in this paper, we compare the outcome of two different clustering algorithms, namely DPClus and Newman algorithms, by applying them to a PPI network, and point out some advantages and limitations of both.

1. Introduction

Research on complicated networks has become very popular and wide spread with recent advancement of computers. There are various types of networks, for example, world wide web, Protein-Protein Interaction (PPI) networks, gene networks, food webs, citation networks, social networks and so on[1)~6)]. These networks are usually studied as a graph consisting of a node set and an edge set. In a PPI network, which is one of the subjects of our research, a node corresponds to a protein and an edge corresponds to an interaction. PPI networks are expected to be a strong basis for prediction of protein functions. For the purpose of analyzing the structures of these networks, it is important to develop good network-visualizing methods. We have been developing the visualizing tool based on clustering where clusters are defined as groups of relatively densely connected nodes in the network[6)]. Some researchers have studied about sensuousness of network visualizations, for example, Batini[7)] defined some sensuousness rules of network visualizing. However, it is difficult for us to consider sensuousness for complicated networks when we visualize them, because they have many nodes and edges. If we try to visualize a large scale network entirely on a display, it will usually end in failure. The visualizing network will look like a square which are covered with nodes and edges, because of abounding number of nodes and edges. This kind of visualizing is not so good and it is hard to understand any network structure visually in this way. Given the fact that visualization of an entire network is difficult, it is natural that we must focus on "partial visualization". In this paper, we address "cluster" as the target of "partial visualization". Therefore, we proposed a comprehensive visualizing method on the basis of cluster structure of networks. In the present study, clustering means dividing node set into cohesive groups such that the nodes of a group are relatively densely connected in the graph as shown in **Fig. 1**.

In this work, we used the algorithms of Refs. 7) and 8). The algorithm of Ref. 7) is based on the concepts of density and periphery of clusters and will be referred to as DPClus algorithm. On the other hand, the algorithm of Ref. 8) uses global information about network structure referred to as Newman algorithm. Clustering approaches of these two algorithms are quite different, so even if we give the same network data to these algorithms, we will not get the same output. So far it is difficult to strictly define a complex/cluster/module regarding a network that can satisfy all purposes. As a result many clustering algorithms have been developed in view of different aspects. We compare the outcome of two different clustering

† Department of Bioinformatics and Genomics, Graduate School of Information Science, Nara Institute of Science and Technology
†† Department of Computational Biology, Graduate School of Frontier Sciences, the University of Tokyo

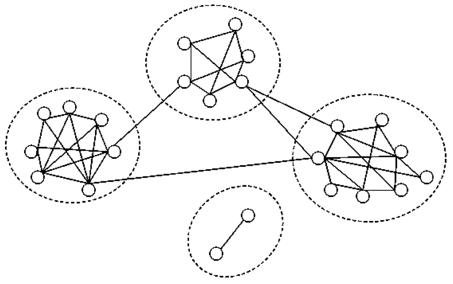


Fig. 1 Notion of clusters in a network. The clusters are represented by the dotted lines.

algorithms by applying them to a PPI network, and point out some advantages and limitations of both. It seems a better algorithm can be developed by considering the positive points of these two. In addition, we used PPI network as subject of visualization, because we thought that there is close relation between clustering and prediction of protein functions, and because if we can visualize the PPI network entirely, it can be expected that we can predict unknown protein functions visually. All interaction data are represented as binary relations between protein pairs and can be easily regarded as a network. In the present paper, we also briefly discuss GRINEditC which is a network visualizing software tool. This is an upgraded version of GRINEdit[9] which has been developed previously in our laboratory for visualizing complicated networks. One of the motivations of the present work is to find out a suitable clustering algorithm for GRINEditC. In the following section we introduce the DPClus and Newman algorithms. In the section entitled "GRINEditC", we briefly discuss our visualization software. In the Results and Discussion section, we compare the outcome of clustering by applying DPClus and Newman algorithms to a PPI network. Finally as conclusion, we collect up the important points of this paper.

2. Clustering Algorithms

In this section, we introduce two clustering algorithms which we used to visualize PPI networks.

2.1 DPClus Algorithm

In DPClus, a network is considered as an undirected simple graph G = (N, E) that consists of a finite set of nodes N and a finite set of edges E. Before details of the algorithm, we define some terminologies used in this algorithm.

Definition 1. Density d_k of any cluster k is the ratio of the number of edges present in the cluster ($|E_k|$) and the maximum possible number of edges in the cluster ($|E_k|_{max}$) and is represented by Eq. (1).

$$d_k = \frac{|E_k|}{|E_k|_{max}} = \frac{2 \times |E_k|}{|N_k| \times (|N_k| - 1)} \quad (1)$$

Here, $|N_k|$ is the size of the cluster, i.e., the number of nodes in the cluster. The density of a cluster is a real number ranging from 0 to 1.

Definition 2. The **cluster property** cp_{nk} of any node n with respect to any cluster k of density d_k and size $|N_k|$ is defined by Eq. (2).

$$cp_{nk} = \frac{|E_{nk}|}{d_k \times |N_k|} \quad (2)$$

Here, $|E_{nk}|$ is the total number of edges between the node n and each of the nodes of cluster k. In **Fig. 2** (a), the cluster property of node f with respect to cluster $\{a, b, c, d, e\}$ is $2/(0.7 \times 5) \approx 0.57$ while in Fig. 2 (b) the cluster property of node f with respect to cluster $\{a, b, c, d, e\}$ is $1/(1 \times 5) = 0.2$. A higher value of cluster property of a neighbor indicates that it is part of the cluster while a lower value indicates that it is part of the periphery. The graph of Fig. 2 (b) can be separated into two clusters by using the concept of cluster property.

Definition 3. The **weight** w_{uv} **of an edge** $(u, v) \in E$ is the number of the common neighbors of the nodes u and v.

Definition 4. The **weight** w_n **of a node** n is the sum of the weights of the edges connected to the node i.e. $W_n = \sum W_{nu}$ for all u such that $(n, u) \in E$.

In the previous paper we discussed details of the algorithm[7]. So here we describe it somewhat briefly. The flowchart of the algorithm is shown in Fig. 2 (d) and it is divided into five major steps: Input & initialization, Termination check, Seed selection, Cluster formation and Output & update.

Input & initialization: The input to the algorithm is an undirected simple graph and hence the associated matrix of the graph is read first. It is also necessary to provide a value of minimum density we allow for the generated clusters and a minimum value for cluster property that determines the nature of periphery tracking. From now on, these input values of density and cluster property will be referred to as d_{in} and cp_{in} respectively. Clustering can be performed several times using different input values for d_{in} and cp_{in},

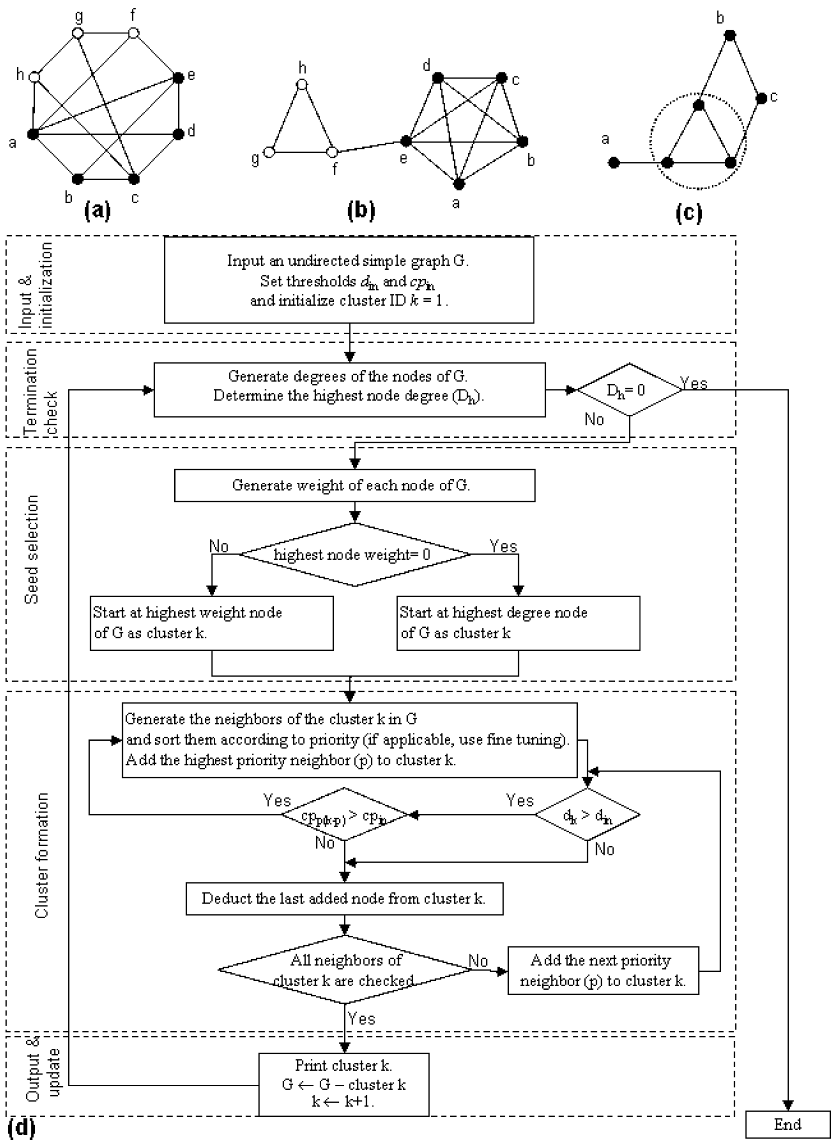

g
f
h
e
a
d
b
c
(a)
h
d
c
g
f
e
b
a
(b)
b
c
a
(c)
Input & initialization
Input an undirected simple graph G.
Set thresholds d_{in} and cp_{in}
and initialize cluster ID k = 1.
Termination check
Generate degrees of the nodes of G.
Determine the highest node degree (D_h).
D_h= 0
Yes
No
Seed selection
Generate weight of each node of G.
highest node weight= 0
No
Yes
Start at highest weight node of G as cluster k.
Start at highest degree node of G as cluster k
Cluster formation
Generate the neighbors of the cluster k in G and sort them according to priority (if applicable, use fine tuning).
Add the highest priority neighbor (p) to cluster k.
Yes
$cp_{p(k+p)} > cp_{in}$
No
Yes
$d_k > d_{in}$
No
Deduct the last added node from cluster k.
All neighbors of cluster k are checked
No
Yes
Add the next priority neighbor (p) to cluster k.
Output & update
Print cluster k.
G ← G − cluster k
k← k+1.
End
(d)

Fig. 2 Concepts of the DPClus algorithm. (a) and (b) Two typical graphs of the same size and density. (c) A typical cluster and its neighbors. (d) Flow-chart of the algorithm.

which allows the suitable set of clusters to be chosen from among a number of options. The cluster ID, k is initialized to 1.

Termination check: Once a cluster is generated, it is removed from the graph. The next cluster is then formed in the remaining graph and the process goes on until no edge is left in the remaining graph. For a graph with no edge, the degree of each node is zero. When such situation arrives, the algorithm terminates.

Seed selection: Each cluster starts at a deterministic single node which we call the seed node. The highest weight node is considered as the seed node. However, if the highest node-weight is zero, the highest degree node is considered as the seed node. The weights of nodes are determined by summing up the weights of incident edges and the weights of edges are calculated by matrix multiplication.

Cluster formation: The cluster starts as a single node and then grows gradually by adding nodes one by one from its neighbors. The neighbors of a cluster are the nodes connected to any node of the cluster but not part of the cluster. It is very important to add priority neighbors to the cluster first to guide the cluster formation in a proper way. The priority is determined based on two measures: (1) the sum of the weights of the edges between a neighbor and each of the nodes of the cluster and (2) the number of edges between a neighbor and each of the nodes of the cluster. Therefore, a double sorting is performed to sort the neighbors. Furthermore, we use some fine-tuning in the sorting process when cluster length is more than one but all neighbors are connected to the cluster by only single edge. In the following example we explain the purpose of fine-tuning. Fig. 2 (c) shows a dotted-line encircled cluster, say at some instant of the cluster formation process, and its neighbors a, b and c. All three neighbors are of equal priority if sorting is performed according to the two measures mentioned above. However, by common sense we realize that b or c should be given more priority. The fact is that other than the single edge link with the cluster, b or c is also connected to the cluster by a link of length 2 that goes outside of the cluster. Based on this fact, we fine-tune the sorting of the neighbors such that b or c comes up as the highest priority neighbor. However when fine-tuning is used to sort the neighbors, we use half the value of cp_{in} for periphery checking and thus help to form some sparse clusters. We check two things before adding a node to a cluster. First, we make sure that addition of the node to the cluster does not cause the density d_k of the cluster to fall below d_{in}, the input density. Second, we check whether the node is part of the cluster or part of the periphery. If a node is part of the cluster it should be connected to a reasonable number of edges within the cluster. For example for a cluster of density d_k, each node on an average should be connected to $d_k \times (|N_k| - 1)$ edges within the cluster, where $|N_k|$ is the size of the cluster. We do not add a neighbor to a cluster if its cluster property is less than cp_{in}. We can choose the value of cp_{in} from within the range $0 < cp_{in} < 1$.

Output & update: Once a cluster is generated, it is printed and graph G is updated by removing the present cluster, i.e. the nodes belonging to the present cluster and the incident edges on these nodes are removed from G. The cluster ID, k is updated by adding 1 to it.

2.2 Newman Algorithm

In this section, we briefly introduce Newman algorithm[8] for network clustering. Newman's method is based on modularity, and proposes the parameter Q-value as a measure of network modularity. First, this method considers each node as a cluster. In each subsequent step Newman's method combines two clusters for which the increment of Q-value is the maximum and hence this is a greedy algorithm. The process of combining goes on until Q-value can not be increased further. Q-value is defined as follows:

$$Q = \sum (e_{ii} - a_i^2) \tag{3}$$

Here, e_{ii} means the fraction of edges in cluster i with respect to all the edges in the network and a_i means the fraction of the number of edges that end in cluster i. If we try to calculate all cases of Q, we need to spend more than exponential order of time[8]. However, we can avoid this computational burden because we only need to calculate the increment ΔQ when we combine two clusters, say cluster i and cluster j. Newman showed that ΔQ can be calculated as follows:

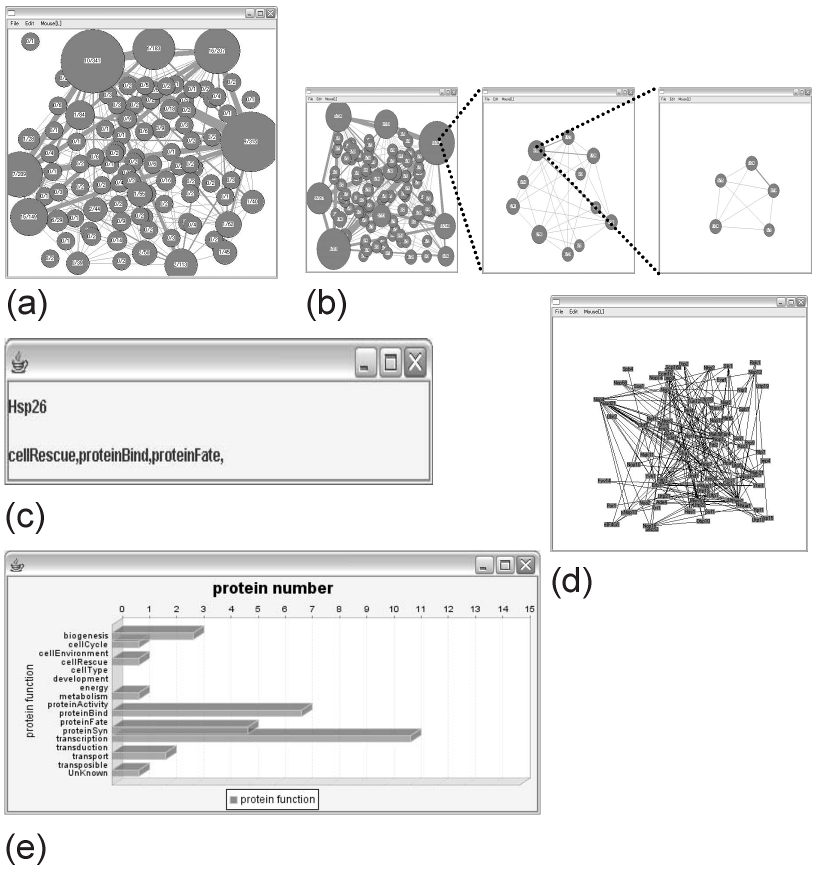
$$\Delta Q = e_{ij} + e_{ji} - 2a_i a_j = 2(e_{ij} - a_i a_j) \tag{4}$$

Here, e_{ij} means the half of the fraction of edges between cluster i and cluster j with respect to all the edges in the network. Complexity of calculating ΔQ is of the order $O(N)$ where N is the number of nodes in the network. We combine two clusters at most N-1 times, so in sparse networks, we can complete clustering in $O(N^2)$ time.

3. GRINEditC

GRINEditC is the software which can visualize large networks with the aid of clustering in a hierarchical fashion. In GRINEditC we consider a network as a simple graph, i.e., we do not consider the direction of edges, self-loop and multiple-edges between nodes. This program was written with Java and the software has now been customized for the purpose of visualizing PPI networks so that we can analyze these networks with ease. In the following we explain the aspects of this software.

(1) GRINEditC reads "input network" from a file which contains relations between arbitrary two proteins and additional information about proteins.
(2) GRINEditC can execute DPClus cluster-



Hsp26

cellRescue,proteinBind,proteinFate,

Fig. 3 Features of GRINEditC.

ing and Newman clustering upon an input network and can visualize the cluster-graph in a window (**Fig. 3** (a)), where each node is a cluster. The visualizing size of a cluster-node is determined in proportion to the number of nodes in the corresponding cluster. The visualizing width of an edge in the cluster-graph is determined depending on the number of edges between the nodes in two clusters.

(3) If a user selects menu item "reclustering" and clicks on a cluster, Newman clustering algorithm will be applied to this cluster, and the corresponding cluster-graph is visualized in another window. We can do "reclustering" as long as the target cluster contains more than two nodes (Fig. 3 (b)).

(4) If a user chooses a cluster that has only one node, and selects "reclustering", he/she can see a popup window which contains information on this single protein, its name and functions (Fig. 3 (c)).

(5) If a user selects "Show subgraph" menu, chooses one cluster-node by clicking, he/she can see subgraph under this cluster in a separate window (Fig. 3(d)).

(6) If a user selects "Show rate chart" menu, chooses one cluster-node by clicking, he/she can see distribution of the proteins of the cluster with respect to functional classes (Fig. 3(e)).

4. Results and Discussion

As far as we know, there is no algorithm that always outputs optimal clustering results. Therefore, it is reasonable to compare the outputs of different algorithms and pointing out the advantages and disadvantages of each. This may lead to invention of better algorithm by way of adding up the advantages of several algorithms. So we compare DPClus algorithm with Newman algorithm on the basis of clusters they generate from a PPI network. DPClus algorithm and Newman algorithm are clustering algorithms, but their clustering approaches are very different from each other. For the sake of comparison, we applied both of them to $S.$

cerevisiae PPI data from DIP[10]. In case of DPClus algorithm, we used 0.6 as input density and 0.5 as input cluster property. We performed Newman clustering based on maximizing the Q-value following a greedy approach. In the following, we compare the results of clustering by these two algorithms from various perspectives.

4.1 Comparison Based on Size of Clusters

DPClus and Newman algorithm extracted 167 and 78 clusters respectively from the *S. cerevisiae* PPI network of DIP[10]. The size of the largest fifty clusters generated by each of the algorithm is plotted in **Fig. 4**. The size of the biggest cluster generated by DPClus is 33 while that in case of Newman algorithm is 987. From Fig. 4, it is evident that some clusters generated by Newman algorithm are too big while the others are too small depicting some imbalance in the size of the generated clusters. On the other hand, a few clusters generated by DPClus algorithm are somewhat big and others are of balanced size.

4.2 Comparison of Density Distrubutions

Similar to the previous section, we considered the largest fifty complexes generated by each of the two algorithms and we show the distribution of these complexes with respect to density in **Fig. 5**. Let x be the label of a column. A cluster whose density d is within the range $x \leq d < x + 0.1$ has been counted for the column with label x. While clustering using DPClus algorithm, we selected input density as 0.6 and hence all the clusters generated by DPClus algorithm have density 0.6 or more. On the other hand, in case of the clusters generated by Newman algorithm, some clusters are

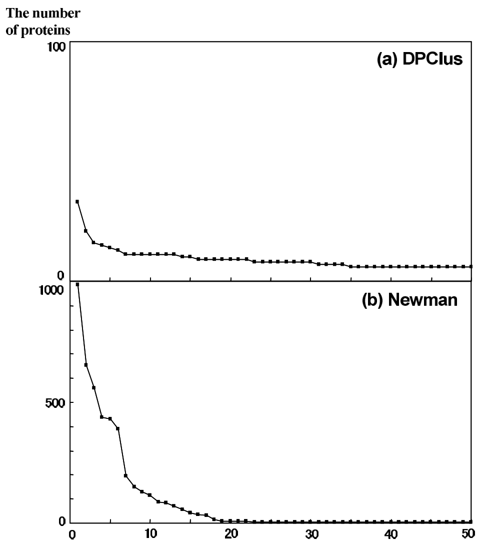


Fig. 4 Sizes of the largest fifty clusters generated by DPClus and Newman algorithms.

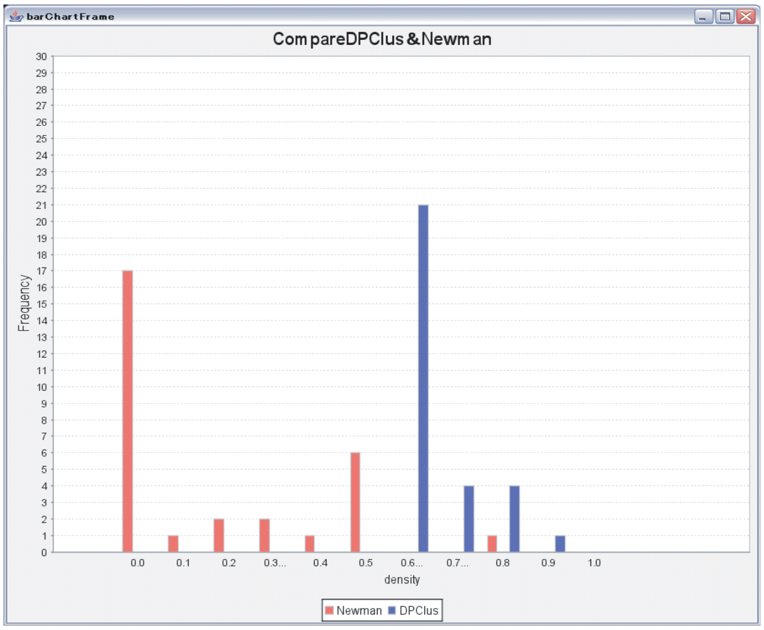


Fig. 5 Distribution of clusters with respect to density.

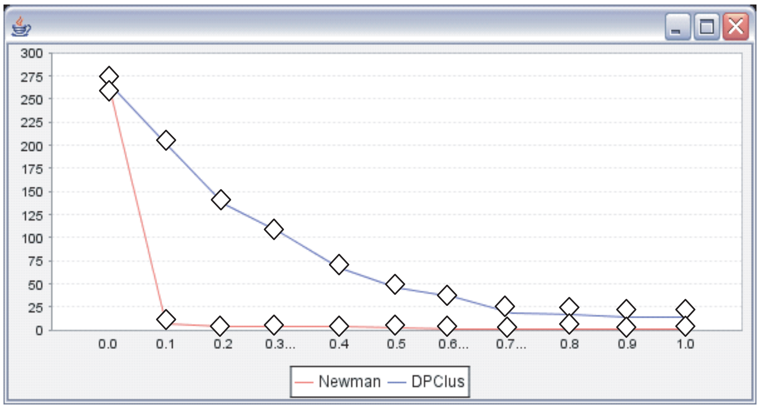


Fig. 6 Cumulative distribution of known complexes with respect to ω.

of high density, but most of the clusters are of low density. Density can be regarded as an overall measure of cohesiveness of the nodes of a cluster. So DPClus makes it possible to select more cohesive clusters compared to those of Newman algorithm.

4.3 Matching with known complexes

We also investigate how the generated clusters by these two algorithms fit with the known protein complexes. We collected 267 known protein complexes from MIPS [11]. To calculate how effectively a predicted complex matches or overlaps with a known complex, we use a measure ω defined as follows:

$$\omega = \frac{i^2}{a \times b} \tag{5}$$

Here, i is the size of the intersection set of a predicted complex with a known complex, a is the size of the predicted complex and b is the size of the known complex. We determined how the known complexes matched with complexes predicted by both the DPClus algorithm and the Newman algorithm. We relate or assign a known complex to a predicted complex if the overlapping score ω between them is the maximum. The cumulative distribution of known complexes with respect to ω is shown on **Fig. 6**. We can easily find that many of the predicted complexes by DPClus algorithm substantially matched with known complexes.

4.4 Comparison of p-value Distributions

The proteins of *S. cerevisiae* can be classified into 16 functional classes [11] as follows: (1) Cell cycle and DNA processing, (2) Protein with binding function or cofactor requirement (structural or catalytic), (3) Protein fate (folding, modification, destination), (4) Biogenesis of cellular components, (5) Cellular transport, transport facilitation and transport routes, (6) Metabolism, (7) Interaction with the cellular environment, (8) Transcription, (9) Energy, (10) Cell rescue, defense and virulence, (11) Cell type differentiation, (12) Cellular communication/signal transduction mechanism, (13) Protein activity regulation, (14) Protein synthesis, (15) Transposable elements, viral and plasmid proteins, and (16) Development (Systemic). To assess the statistical significance of functional richness of proteins in individual clusters we calculated their p-values using the following formula:

$$P = 1 - \sum_{i=0}^{k-1} \frac{\binom{F}{i}\binom{N-F}{C-i}}{\binom{N}{C}} \tag{6}$$

Here N is the number of all proteins in PPI data, C is the number of proteins in the cluster, F is the number proteins of a functional group in the network, and k is the number of proteins of the functional group in the cluster. In general it can be suggested that the lower the p-value the higher the statistical significance of the cluster. Actually, it was difficult to calculate p-value directly and we used the concept of logarithm to calculate them. In **Fig. 7**, we show the distribution of largest fifty complexes generated by each of the two algorithms with respect to their p-values. The upper chart corresponds to Newman algorithm while the lower chart corresponds to DPClus algorithm. We can easily find that the DPClus algorithm performed well compared to Newman algorithm in terms of p-values of the generated clusters.

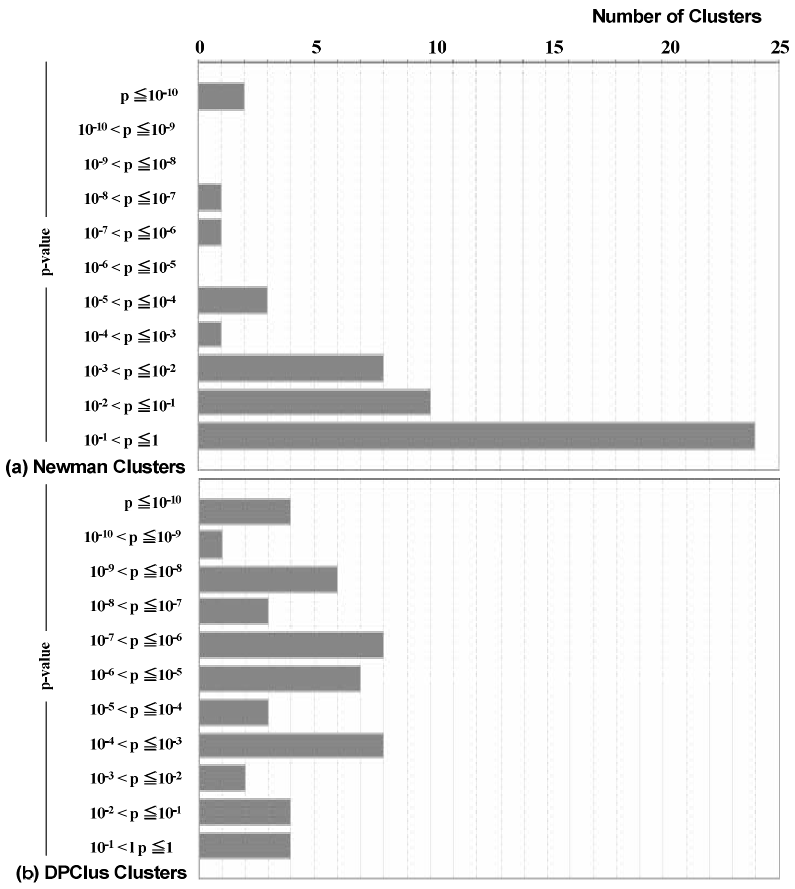


Fig. 7 Distribution of clusters with respect to p-value. The upper chart corresponds to Newman algorithm and the lower chart corresponds to DPClus algorithm.

4.5 Comparison on the Basis of Functional Richness

To predict functions of proteins, it is important to assess how similar function proteins are accumulated in individual predicted clusters. For this purpose, we considered 10 largest clusters predicted by each of the algorithms and sixteen functional classes from MIPS. The names of these sixteen functional classes have been mentioned in the caption of **Fig. 8**. cluster ID to each cluster starting from the biggest one to the decreasing order of their size. We calculated percentage of proteins of each function present in each cluster. The results are shown in Fig. 8, where we show a histogram corresponding to each cluster. The horizontal axis corresponds to 16 functional classes. The heights of the columns of a histogram depict the percentage of proteins of a complex belonging to the corresponding functional classes in the context of the total number of proteins in each cluster. The highest column/columns of a histogram are colored as red. It is evident that the higher the red column in a histogram the better the functional richness of the cluster. The average of the percentages corresponding to the red columns is 0.8602 in case of DPClus algorithm and 0.2738 in case of Newman algorithm. Therefore, we can conclude that DPClus algorithm performed better compared to Newman algorithm in case of extracting clusters that are rich with similar function proteins.

It seems that DPClus algorithm works better than Newman algorithm in the matter of predicting protein complexes from PPI networks. However, as we previously mentioned, the clustering approaches of DPClus algorithm and Newman algorithm are quite different from each other. Roughly speaking, DPClus algorithm focuses on local information like density

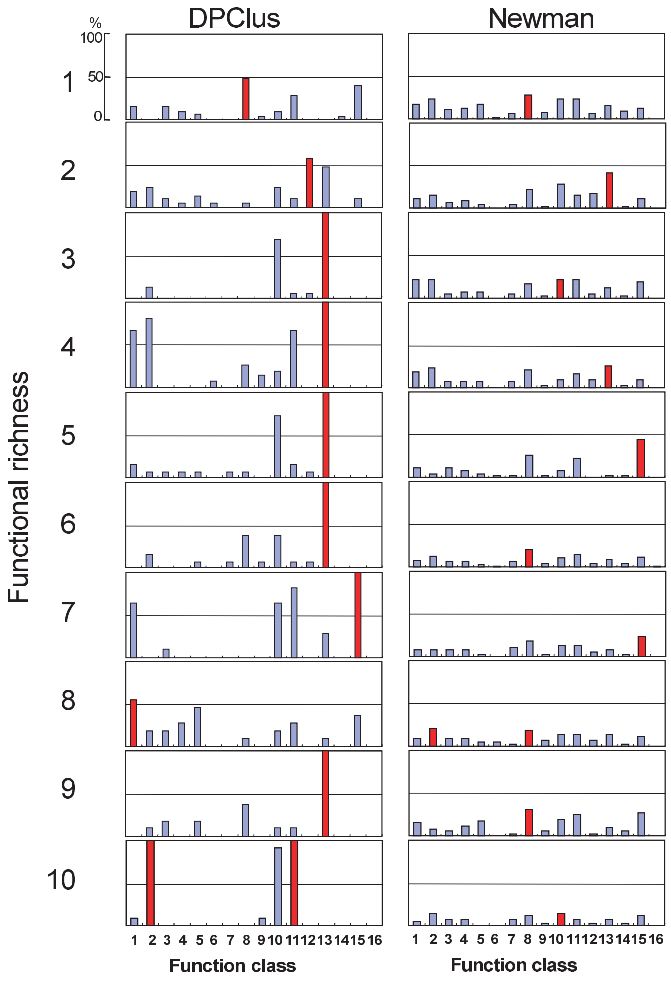


Fig. 8 Distribution of proteins in individual clusters with respect to following functional classes: (1) Cell cycle and DNA processing, (2) Protein with binding function or cofactor requirement (structural or catalytic), (3) Protein fate (folding, modification, destination), (4) Biogenesis of cellular components, (5) Cellular transport, transport facilitation and transport routes, (6) Metabolism, (7) Interaction with the cellular environment, (8) Transcription, (9) Energy, (10) Cell rescue, defense and virulence, (11) Cell type differentiation, (12) Cellular communication/signal transduction mechanism, (13) Protein activity regulation, (14) Protein synthesis, (15) Transposable elements, viral and plasmid proteins, and (16) Development (Systemic).

and cluster property, while Newman algorithm focuses on global information by way of optimizing Q-value. So, Newman algorithm may be better in the sense that it takes into account the global structure of the network. But, Newman algorithm has ambiguities in the early step of clustering, so some roughness could be seen in the result of clustering. On the other hand, DPClus does not focus on any type of global optimization. Introducing some optimizing parameter in case of DPClus algorithm might pave the way of finding a better clustering algorithm.

5. Conclusion

In this paper, we discussed the visualizing method of PPI networks in which the nodes represent proteins and the edges represent interactions between protein pairs. Our visualizing method focuses on clusters, which is defined as relatively densely connected group of nodes in a network. Recounting all clusters from a network is called clustering. Protein functions can be predicted by visualizing a PPI network with the aid of clustering. However, the results of visualization are quite different depending on which clustering algorithm we use. In addition, it seems that there is no optimum clustering algorithm and a better algorithm can be developed by considering the positive points of more than one algorithm. For this purpose, we compared the performances of two clustering algorithms, the DPClus algorithm and the Newman algorithm and point out some advantages and limitations of both.

References

1) Strogatz, S.H.: Exploring complex networks, *Nature*, Vol.410, pp.268–276 (2001).
2) Newman, M.E.J.: The Structure and Function of Complex Networks, *SIAM, Rev.*, Vol.45, No.2, pp.167–256 (2003).
3) Albert, R. and Barabase, A.L.: Statistical mechanics of complex networks, *Rev. Mod. Phys.*, Vol.74, No.1, pp.44–97 (2002).
4) Guimera, R., Danon, L., Diaz-Guilera, A., Giralt, F. and Arenas, A.: Self-similar community structure in a network of human interactions, *Phys. Rev. E*, Vol.68, No.065103 (2003).
5) Wilkinson, D. and Huberman, B.A.: A method for finding communities of related genes, *Proc. Natl. Acad. Sci. U.S.A.*, Vol.101, pp.5241–5248 (2004).
6) Holme, P., Huss, M. and Jeong, H.: Subnetwork hierarchies of biochemical pathways, *Bioinformatics*, Vol.19, No.4, pp.532–538 (2003).
7) Amin, M.A., Shinbo, Y., Mihara, K., Kurokawa, K. and kanaya, S.: Development and implementation of an algorithm for detection of protein complexes in large interaction networks, *BMC Bioinformatics*, Vol.207, No.7, pp.532–538 (2006).
8) Newman, M.E.J.: Fast algorithm for detecting community structure in networks, *Phys. Rev*, Vol.69, No.066133 (2004).
9) GRINEdit: http://www.nishiohirokazu.org/grinedit/.
10) DIP: http://dip.doe-mbi.ucla.edu/.
11) MIPS: http://mips.gsf.de/.

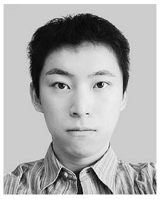
(Received August 25, 2006)
(Accepted September 10, 2006)

(Communicated by *Tatsuya Akutsu*)

Hisashi Tuji received B.S. from University of Gunma in 2005. From 2005, he has been a Master course student at Department of Bioinformatics, Graduate School of Information Science, Nara Institute of Science and Technology.

Md. Altaf-Ul-Amin received B.Sc. in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology (BUET) in 1993, Master of Science in Electrical, Electronic and Systems Engineering from University Kebangsaan Malaysia (UKM) in 1999 and Ph.D. from Nara Institute of Science and Technology (NAIST) in 2003. Presently he is working as an Assistant Professor in Comparative Genomics Lab. of NAIST. His research interest includes application of Network theory and algorithms and self organizing mapping to bioinformatics.

Masanori Arita received Ph.D. from Department of Information Science, Graduate School of the University of Tokyo in 1999. He was Researcher in Electrotechnical Laboratory at AIST 1999–2001 and in Computational Biology Research Center at AIST 2001–2003. Associate Professor in Department of Computational Biology, Graduate School of Frontier Sciences, the University of Tokyo 2003–present.



Hirokazu Nishio received Ph.D. from Nara Institute of Science and Technology in 2006. His research interest includes Visualization of Large-scale data obtained by biological experiment.

Yoko Shinbo received Bachelor of Agriculture in Agricultural Science from Kagoshima University in 2002, M.Sc. in Department of Bioinformatics, Graduate School of Information Science, Nara Institute of Science and Technology (NAIST) in 2004. From 2004, she has been working as research fellow in Comparative Genomics Lab. of NAIST. She is doing research on database construction of secondary metabolite and biosyntheses pathway.

Ken Kurokawa received Ph.D. in Environmental Science & Microbiology from Graduate School of Pharmacy, Osaka University in 1995. M.Sc. in Earth Physics from Institute of Geology and Paleontology, Graduate School of Science, Tohoku University in 1993, B.Sc. in Structural Geology and Volcanology from Institute of Geology and Paleontology, Faculty of Science, Tohoku University in 1990. He was Associate Professor in Bioinformatics at Graduate School of Information Science, NAIST from 2004, Research Associate in Bioinformatics, Research Institute of Microbial Diseases, Osaka University 2001–2004. Post-Doctoral Fellow in Bioinformatics Genome Information Research Center, Osaka University 1998–2001. His interests are origins and evolution of genome organization. Particularly interested in the correlation between genome information and biosystem evolution. He seeks out the laws of self-organization in genome, and dynamic modeling of biosystems with network thermodynamics.

Shigehiko Kanaya received B.S. from Department Bioscience, Faculty of Science, Science Univercity of Tokyo in 1985, Ph.D. from Department Material and System Engineering, Toyohashi University of Technology in 1990. In 1990, Assistant Researcher in Information Engineering at Yamagata University, with a research program on genome informatics concerning species-specific codon usage on the basis of multivariate analysis. In 1996, Guest Associate Professor at National Institute of Genetics. In 1999 Associate Professor, Electronic and Information Engineering, and in 2000 Associate Professor, Applied Biosystem Engineering at Yamagata University. In 2004–present, Professor at Comparative Genomics, Department of Bioinformatics, Graduate School of Information Science, Nara Institute of Science and Technology.