ANNOTATING GENE FUNCTIONS WITH INTEGRATIVE SPECTRAL CLUSTERING ON MICROARRAY EXPRESSIONS AND SEQUENCES

LIMIN LI[1]
limin@hkusua.hku.hk

MOTOKI SHIGA[2]
shiga@kuicr.kyoto-u.ac.jp

WAI-KI CHING[1]
wching@hkusua.hku.hk

HIROSHI MAMITSUKA[2]
mami@kuicr.kyoto-u.ac.jp

[1] *Advanced Modeling and Applied Computing Laboratory, Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong*
[2] *Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji 611-0011, Japan*

Annotating genes is a fundamental issue in the post-genomic era. A typical procedure for this issue is first clustering genes by their features and then assigning functions of unknown genes by using known genes in the same cluster. A lot of genomic information are available for this issue, but two major types of data which can be measured for any gene are microarray expressions and sequences, both of which however have their own flaws. Thus a natural and promising approach for gene annotation is to integrate these two data sources, especially in terms of their costs to be optimized in clustering. We develop an efficient gene annotation method with three steps containing spectral clustering over the integrated cost, based on the idea of network modularity. We rigorously examined the performance of our proposed method from three different viewpoints. All experimental results indicate the performance advantage of our method over possible clustering/classification-based approaches of gene function annotation, using expressions and/or sequences.

Keywords: Gene function annotation; spectral clustering; network modularity; integrative clustering.

1. Introduction

Proteins play important roles in a lot of biological functions in a cell and thus functional annotation of genes is a fundamental problem in the post-genomic era. Even for yeast, one of the most well-studied organisms, about one-fourth of all genes still remain uncharacterized [20]. We address the issue of predicting gene functions based on clustering (or grouping) genes into modules [1, 41]. That is, genes are clustered and then functions of an unknown gene are predicted by using functionally known genes in the same cluster (or module).

Various types of data have been used for predicting gene functions in the literature, including sequences, expressions, protein-protein interactions and phylogenetic profiles, but basically only sequences and microarray expressions are the information which can be measured for any gene. However, a problem is that each of these two information sources has its own flaws. Sequence similarity is, in most cases, correlated to functional similarity, but there exist exceptions. In fact it is reported that even highly aligned sequences have totally different functions in some cases [25]. Microarray expressions are weak at data quality: bad probes can be contained and elements in an expression matrix can be corrupted.

Thus, integrating these two data sources is a natural and promising direction to achieve a better performance for the problem of annotating functions of any gene. However, these two data types are different, and integrating sequences with expressions is not simple. An expression profile is a numerical vector, being uniquely mapped in a space of microarray experiments, which results in that microarray expressions are, in machine learning, *structured data* with examples in rows and features of examples in columns. On the other hand, for sequences, similarities are useful for gene clustering, meaning that this data can be an association (or affinity) matrix between sequences, resulting in a weighted network or graph, which is, in machine learning, *unstructured data*. A possible approach for combining these two data types is "data transformation" between each other. For example, expression profiles can be converted into an association matrix by computing correlations between all possible pairs of expression profiles. Then, we can integrate the expression association with the sequence association to generate a total association matrix, which can be a gene network with association weights on edges. In fact, network-based function prediction has been pronounced recently [27]. The reverse way is also possible. That is, a sequence profile can be generated from each sequence and added to the expression profile of the same gene. Once we make such "joint profiles", a lot of techniques including k-means and hierarchical clustering can be employed for clustering them. Furthermore if we fix the set of gene functions beforehand, the problem can be a multi-class classification problem, and in fact classification techniques such as support vector machine (SVM) have been already applied to gene annotation [6, 7] though the data is not necessarily confined to sequences and expressions. We note that this type of data transformation has problems: First, significant information might be lost, being easily implied by the fact that we cannot reproduce the original data from the transformed data. Another doubt can be casted whether a simple addition of two datasets in the above manner would succeed or not.

Clustering numerical vectors usually results in minimizing some cost function, such as a dissimilarity cost function [17]. Similarly clustering over associations, i.e. a gene network, can be an optimization problem which minimizes some criterion, such as normalized cut over a network [28]. Thus we propose an integrative clustering method which minimizes the total cost obtained by linearly weighting the two clustering costs, which are derived from the two different types of datasets. We emphasize that our cost optimization approach is more natural and adequate than the above data transformation approach, because we can use original data directly without losing any information.

In computer science, a close problem setting of ours is constrained clustering [37, 38], where examples of numerical vectors are clustered with two constraints: "must-link", in which two examples must be in the same cluster, and "cannot-link", in which two examples must be in different clusters. A more similar problem setting is semi-supervised clustering [3], and a typical approach is based on a hidden Markov random field (HMRF) in which variables (or nodes) on a random field are constrained with each other and probabilistic parameters attached at nodes are estimated from examples [3]. However, HMRF has a large number of probability parameters which need a lot of computational cost to estimate. Furthermore, this optimization is likely to be affected by initial values, implying that a very large number of trials are required to find a relatively favorable result. In fact, the most related work to us must be [29], which uses a new hidden random field to be optimized by an EM (Expectation Maximization) algorithm and suffered from the problems as written in the above for HMRF.

Our method, which is based on *spectral clustering* [23, 36], has a clear advantage over the random field-based methods in computational efficiency because the optimization problem is solved by time-efficient matrix computation. In addition, the initial parameter problem of random fields is far relaxed or solved. We further emphasize that spectral clustering is a high-lighted approach in the current machine learning literature as well as a *de facto* standard approach in modern graph partitioning. Furthermore for the clustering criterion to be optimized, our method focuses on the idea of *network modularity* [21, 22]. This is a well-recognized, important network property as well as small-world phenomena [39], scale-free property [2] and self-similarity [32], which are all common to a lot of modern network-shaped data such as world wide web and various biological networks. In this light, our proposed method is a computationally original as well as powerful approach for the gene annotation problem, which is very significant in the post-genome era.

Our whole scheme for gene annotation has three steps: First, from sequence information we generate a gene network in terms of sequence similarity by using a pairwise sequence alignment algorithm. We then do clustering this network, combining with microarray expressions, by using the proposed spectral clustering method based on the idea of network modularity. Finally we assign functions of unknown genes by using the clusters, which were generated by the preceding step.

We evaluated the performance of our scheme in the following three experiments: 1) We first checked the effectiveness of our way of combination of gene expressions and sequences by checking the overlap between predicted clusters and standard functional clusters. 2) We then examined whether our approach of predicting gene functions can outperform other various approaches, particularly simpler data-transformation-based methods, including network-based, k-means-based and SVM-based approaches. 3) We finally validated resultant annotation on unknown genes in Saccharomyces Genome Database (SGD) [20] by using the information of Comprehensive Yeast Genome Database (CYGD) [14].

This paper is organized as follows: Section 2 shows our proposed method which has three steps: 1) generating a gene network by sequence similarity, 2) clustering nodes of a gene network on sequence similarities with microarray expressions of genes, each of which corresponds to a node of the network, and 3) predicting functions of unknown genes by gene functions corresponding to clusters generated by the preceding step. The second step of our method is based on our past work [30], where we proposed a new, general method for clustering numerical vectors considering their network constraints, and verified the performance by basically using a variety of synthetic data. On the other hand, this paper focuses on the problem of annotating gene functions, particularly using the data of microarray expressions and a gene network on sequence similarities. In fact our method in this paper has three steps, and only the second step is related with [30]. Furthermore, we extensively examined the performance of our method by using real data of gene expressions and sequence similarities. We emphasize that these points make this paper significantly different from [30]. Section 3 shows our experimental results which can be mainly divided into four parts: 1) generating a new gene network, 2) examining our integrative clustering method, 3) checking the predictive performance of our entire method and 4) annotating unknown yeast genes. Section 4 summarizes the proposed approach and discusses the future work on this method.

2. Proposed Method

Figure 1 shows our entire framework with three steps for predicting gene functions based on integrative spectral clustering. We describe the detail of each step in this section.

2.1. *Notation*

Our method has three inputs. The first input is a table of gene expression profiles. Let X be an input profile table, i.e. $X = [x_1, \ldots, x_N]$, where x_i $(= [x_{i1}, \ldots, x_{ip}]^T)$ is an expression profile for gene i and x_{ij} is an expression value of gene i in sample j. Let $\tilde{x}_i$ be the normalized vector of x_i satisfying with $\tilde{x}_i^T \tilde{x}_i = 1$, and $\tilde{X}$ be the normalized matrix of X $(\tilde{X} = [\tilde{x}_1, \ldots, \tilde{x}_N])$. Let Y be the inner-product of $\tilde{X}$ $(Y = \tilde{X}^T \tilde{X})$. The second input of our method is a gene network, having a gene as a node. For a given network, let w_{ij} be a numerical weight between nodes i and j, taking a range between zero and one. If an input network is a binary network, w_{ij} takes one if there is an edge between i and j; otherwise zero. Let W be a matrix with w_{ij} for element (i, j). Let $d_i = \sum_{j=1}^{N} w_{ij}$ and $D := d^T d$ where $d := (d_1, \cdots, d_N)$. Let D_d be a $N \times N$ diagonal matrix whose (i, i) entry is d_i. The third input of our method is K, which is the number of clusters.

Let z_{ki} be a binary cluster indicator taking one if gene i is in cluster k; otherwise zero. Let z_k be a vector of the k-th cluster assignment: $z_k = [z_{k1}, \ldots, z_{kN}]^T$, and Z be a matrix of cluster assignments $(Z = [z_1, \ldots, z_K])$. Let μ_k be the center of cluster k, and μ be a set of cluster centers $(\mu = [\mu_1, \ldots, \mu_K])$. Let $\mathrm{tr}(A)$ be the trace of

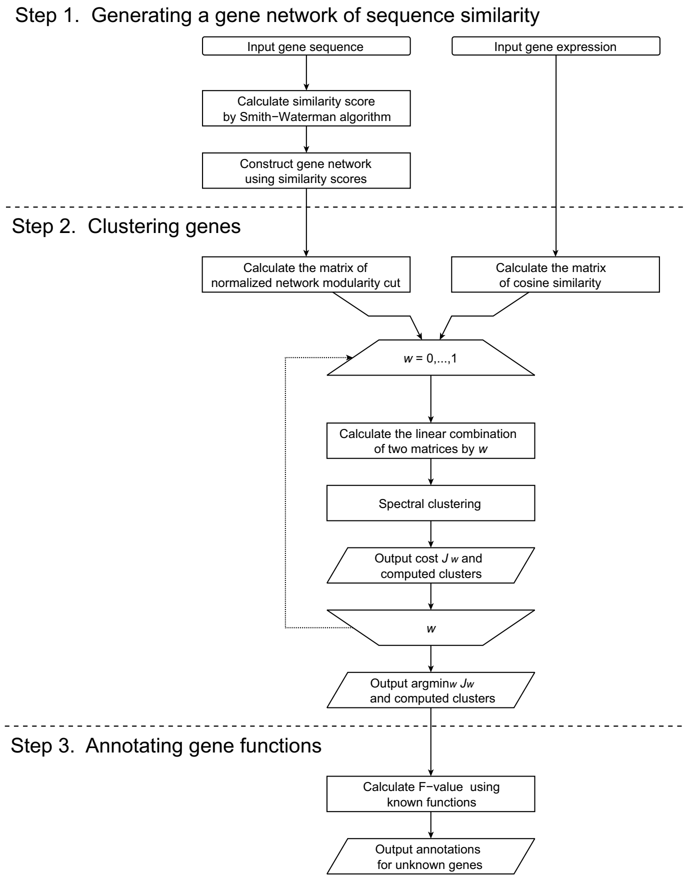

Fig. 1: Entire procedure of our method of annotating gene functions.

matrix A ($\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii}$). Let $L(z_k, z_{k'})$ be the number of edges between clusters k and k', and L be the number of edges in a network: $L = L(\mathbf{1}, \mathbf{1})$, where $\mathbf{1}$ be the N-dimensional column vector with all elements of one. Let N_C be the number of genes in cluster C, and $N_{C,F}$ be the number of genes which are in both clusters C and F.

2.2. *Step 1: Generating a Gene Network of Sequence Similarities*

As shown in our notation of w_{ij}, our framework can handle a real-valued association matrix or a weighted network, but we use a binary network for sequence similarity in our experiments because of computational efficiency and network reliability. We first align all pairs of sequences using the Smith-Waterman algorithm [31] and obtain a list of sequence pairs with similarity p-values, which are a measure indicating how likely each pairwise alignment can be arised by chance. That is, the smaller the similarity p-value is, the more similar the sequence pair is. As the sequence similarity is correlated to the functional similarity, two genes with a low p-value is supposed to share the same function. We place a cut-off value for the p-values to generate a binary affinity matrix between genes, i.e. a gene network. Concretely, a network is generated by drawing an edge between two nodes if the p-value of the pair corresponding to the two nodes is lower than some specified cut-off value. We call this network a *sequence network*.

To find a better cut-off value, we use some standard functional clusters, to which known genes are assigned correctly. We then change cut-off value v against similarity p-values and compute precision, Prec_v, as follows:

$$\mathrm{Prec}_v = \frac{B_v}{A_v},$$

where A_v is the number of gene pairs with similarity p-values less than v, and B_v is the number of gene pairs with similarity p-values less than v and with shared functions. Note that $B_v \subset A_v$. We select an appropriate cut-off value to generate a large sequence network with a high precision and use this cut-off value throughout all experiments.

2.3. *Step 2: Clustering Genes by Combining a Sequence Network with Gene Expressions*

We first define two clustering costs: the cost of clustering numerical vectors and that of clustering nodes over a sequence network. We then linearly combine these two costs and attempt to optimize the total cost by spectral clustering, which transforms the optimization problem into an eigenvalue problem using Lagrange multipliers.

2.3.1. *Two Clustering Costs*

The cost of clustering numerical vectors, i.e. microarray expression profiles, can be defined by the k-means clustering algorithm as follows:

$$J_{num}(X, Z; \mu) = \frac{1}{N} \sum_{k=1}^{K} \sum_{i|z_{ki}=1} \mathbf{Dist}(x_i, \mu_k), \tag{1}$$

where $\mathbf{Dist}(x_i, \mu_k)$ is a distance between numerical vector (expression profile) x_i and cluster center μ_k, borrowing the idea of cosine similarity:

$$\mathbf{Dist}(x_i, \mu_k) = \frac{1}{2}\left(1 - \frac{x_i^T \mu_k}{\sqrt{x_i^T x_i}}\right), \tag{2}$$

and cluster center μ_k can be written as follows:

$$\mu_k = \frac{1}{|z_k|} \sum_{j \in z_k} \tilde{x}_j.$$

By using Eq. (2), Eq. (1) can be transformed into the trace optimization problem as follows:

$$\begin{aligned}
J_{num}(\tilde{X}, Z; \mu) &= \frac{1}{2N} \sum_{k=1}^{K} \sum_{i \in z_k} \left(1 - \tilde{x}_i^T \mu_k\right) \\
&= \frac{1}{2N} \sum_{k=1}^{K} \sum_{i \in z_k} \left(1 - \tilde{x}_i^T \frac{1}{|z_k|} \sum_{j \in z_k} \tilde{x}_j\right) \\
&= \frac{1}{2} - \frac{1}{2N} \sum_{k=1}^{K} \frac{1}{|z_k|} \sum_{i \in z_k} \sum_{j \in z_k} \tilde{x}_i^T \tilde{x}_j \\
&= \frac{1}{2} - \operatorname{tr}\left(\frac{Z^T (2N)^{-1} Y Z}{Z^T Z}\right). \qquad (3)
\end{aligned}$$

Graph partitioning needs some criterion for grouping clusters. The simplest criterion is *graph min-cut*, which minimizes the number of inter-cluster edges, i.e. the number of total edges from a cluster to other clusters, but is likely to generate small clusters which in some cases correspond to outliers. To overcome this shortcoming of graph min-cut, graph min-cut has been normalized in some ways such as *ratio-cut* [8, 15], *min-max cut* [9] and *normalized cut* [28]. In particular, a *de facto* standard criterion for k-way graph partitioning is normalized cut, which is obtained by dividing graph min-cut by the number of edges at each cluster.

In this paper, as the graph cut criterion, we employ the idea of *network modularity* [13, 21, 22], which has been paid a lot of attentions as an important network property in the recent network science. In fact, this property has been found in modern network-shaped data such as metabolic networks [12, 24] and is closely related with network clustering. The network modularity is defined as follows:

$$Q(W, Z) = \sum_{k=1}^{K} \left\{ \frac{L(z_k, z_k)}{L} - \left(\frac{L(z_k, \mathbf{1})}{L}\right)^2 \right\}.$$

As well as graph min-cut, the issue of the cluster size, i.e. the outlier problem, is unconsidered by the original network modularity. Thus we define the new graph-cut criterion, *normalized network modularity cut*, and the cost of graph partitioning (or clustering genes in a given sequence network) of this cut can be given as follows:

$$J_{net}(W, Z) = \sum_{k=1}^{K} \frac{N}{\sum_i z_{ki}} \left\{ \left(\frac{L(z_k, \mathbf{1})}{L}\right)^2 - \frac{L(z_k, z_k)}{L} \right\}$$

We can further transformed the new clustering criterion into a trace optimization framework, borrowing the idea of [40] on the (original) network modularity, as follows:

$$\begin{aligned}
J_{net}(W,Z) &= \sum_{k=1}^{K} \frac{N}{\sum_i z_{ki}} \left\{ \left(\frac{\sum_{i,j} w_{ij} z_{ki}}{L} \right)^2 - \frac{\sum_{i,j} w_{ij} z_{ki} z_{kj}}{L} \right\} \\
&= \sum_{k=1}^{K} \frac{N}{\sum_i z_{ki}} \left\{ \left(\frac{d^T z_k}{L} \right)^2 - \frac{z_k^T W z_k}{L} \right\} \\
&= \sum_{k=1}^{K} \frac{N}{\sum_i z_{ki}} \left\{ \frac{z_k^T d d^T z_k}{L^2} - \frac{z_k^T W z_k}{L} \right\} \\
&= \mathrm{tr} \left(\frac{Z^T N(\frac{1}{L^2} D - \frac{1}{L} W) Z}{Z^T Z} \right) \qquad (4)
\end{aligned}$$

2.3.2. *Optimizing Total Cost*

From the obtained two costs in the preceding section, one can easily see that these two costs share the same denominator. Thus, we combine the above two costs in trace optimization linearly by using weight ω, which is a control parameter taking a real value in the range from zero to one, for balancing the two costs:

$$\begin{aligned}
J_{total}(\tilde{X}, W, Z) &= \omega J_{net}(W,Z) + (1-\omega) J_{num}(\tilde{X}, Z) \\
&= \mathrm{tr} \left(\frac{Z^T (\frac{\omega N}{L^2} D - \frac{\omega N}{L} W - \frac{1-\omega}{2N} Y) Z}{Z^T Z} \right)
\end{aligned}$$

We can easily see that only microarray expressions are considered for ω=0 (denoted by ω_0 in our experiments), while only a given network is used for ω=1 (denoted by ω_1 in our experiments). Our strategy is to solve this trace optimization problem under the constraint of the indicator matrix Z. We first, by relaxing the binary cluster indicators into real values, formulate the clustering problem as the following optimization problem:

$$\text{minimize } \ \mathrm{tr} \left(Z^T (\frac{\omega N}{L^2} D - \frac{\omega N}{L} W - \frac{1-\omega}{2N} Y) Z \right) \quad \text{s.t.} \quad Z^T Z = I_K, \qquad (5)$$

where I_K is the identity matrix of size K. In our experiments, because of practical stability as implied by [40], we used $Z^T D_d Z = I_K$ instead of $Z^T Z = I_K$. Then this optimization problem can be transformed into the following eigenvalue problem by using Lagrange multipliers:

$$M_\omega Z = Z\Lambda,$$

where $M_\omega = D_d^{-\frac{1}{2}} (\frac{\omega N}{L^2} D - \frac{\omega N}{L} W - \frac{1-\omega}{2N} Y) D_d^{-\frac{1}{2}}$. Let U $(= [u_1, \ldots, u_{K-1}] \in \mathbb{R}^{N \times (K-1)})$ denote the matrix containing $K-1$ eigenvectors of M_ω, corresponding to the first $K-1$ smallest eigenvalues. Finally k-means can be applied to the normalized N row vectors of U to obtain gene clusters to be output.

This flow of our method exactly follows spectral clustering, a typical k-way graph partitioning approach, which is a trace optimization with constraints. For example,

Input : $X,\ W,\ K,\ Z^{(0)},\ \mu^{(0)}$
Output : Z

$\tilde{X} \leftarrow X/\sqrt{X^T X}$
$Y \leftarrow \tilde{X}^T \tilde{X}$
Compute D_d and D from W
for $\omega = 0$ to 1 **do**
  $M_\omega \leftarrow D_d^{-\frac{1}{2}}(\frac{\omega N}{L^2}D - \frac{\omega N}{L}W - \frac{1-\omega}{2N}Y)D_d^{-\frac{1}{2}}$.
  Compute U of M_ω.
  Normalize U into $\tilde{U}$.
  $[Z_\omega, \mu_\omega, J_\omega] \leftarrow$ $\mathbf{k\text{-}means}(\tilde{U}, K, Z^{(0)}, \mu^{(0)})$
end for
$\omega_{\min} \leftarrow \arg\min_\omega J_\omega$
$Z \leftarrow Z_{\omega_{\min}}$

Fig. 2: Pseudocode of the clustering algorithm in Step 2.

the cost by the normalized cut can be the following trace:

$$\sum_k \frac{L(z_k, \mathbf{1} - z_k)}{L(z_k, \mathbf{1})} = \sum_k \frac{z_k^T (D_d - W) z_k}{z_k^T D_d z_k} = \mathrm{tr}\left(\frac{Z^T (D_d - W) Z}{Z^T D_d Z}\right). \tag{6}$$

Graph partitioning with normalized cut can be then the following optimization problem:

$$\text{minimize} \quad \mathrm{tr}(Z^T (D_d - W) Z) \quad \text{s.t.} \quad Z^T D_d Z = I_K \tag{7}$$

We note that this trace optimization manner is basically the same as that of ours. However precisely, one can see the difference between our optimization problem and the above (normalized cut-based) problem by examining Eq. (5) and Eq. (7). That is, the above problem can be the so-called *general* eigenvalue problem. Thus an important point is that in principle the normalized cut (as well as the original network modularity) cannot be used to combine numerical vectors with a network, although the normalized cut is a popular criterion in graph partitioning. More precisely, one can see the difference by examining the denominators of the costs. That is, the denominator of the cost of numerical vectors is $Z^T Z$ (Eq. (3)), which is the same as that by normalized network modularity (Eq. (4)) but not that by normalized cut (Eq. (6)). This indicates that numerical vectors and a network can be combined by using the normalized network modularity cut which we propose.

A further key feature of our integrative approach is that we can select the optimal value of parameter ω, which takes a balance between the cost of numerical vectors and that of a network, by choosing the ω which can give the minimum cost in the final k-means step in the eigenspace. This cost minimization is to select the eigenspace in which the clusters are separated the most from each other, and this

Input : $X,\ K,\ Z^{(0)},\ \mu^{(0)}$
Output : $Z,\ \mu,\ J$

k-means $(X, K, Z^{(0)}, \mu^{(0)})$
$Z \leftarrow Z^{(0)},\ \mu \leftarrow \mu^{(0)}$
$\tilde{X} \leftarrow X/\sqrt{X^T X}$
while $J(\tilde{X}, Z; \mu)$ is not converged **do**
  $\mu_k^{(t+1)} \leftarrow \frac{1}{|Z_k^{(t)}|} \sum_{j \in Z_k^{(t)}} \bar{x}_j \quad (k = 1, \cdots, K)$
  $Z^{(t+1)} \leftarrow \arg\min_Z J(\tilde{X}, Z; \mu^{(t+1)})$
  $Z \leftarrow Z^{(t+1)},\ \mu \leftarrow \mu^{(t+1)},\ J \leftarrow J(\tilde{X}, Z; \mu)$
end while

Fig. 3: Pseudocode of k-means.

eigenspace must be the best for clustering genes. We use this optimal ω to have gene clusters which are used for predicting functions of genes.

Figure 2 shows a pseudocode of our entire algorithm of integrative clustering with normalized network modularity, and Figure 3 shows a pseudocode of k-means which is used in Figure 2.

2.4. *Step 3: Predicting Functions of Unknown Genes*

We first assume that we have a set of standard functions (or functional modules) for which known genes are already assigned correctly. We note that any set can be used for our method. In this setting, our annotation strategy, which assumes that genes in the same cluster share the same function, has two steps. 1) We first examine the relevance between each of computed clusters and each of standard functions by using some measure. 2) We then, for each cluster, rank the functions according to the relevance, and the most relevant function is assigned to genes in this cluster as their functions. The problem is how to compute/score the relevance between a computationally obtained cluster and each of standard functions.

We first introduce our default measure, which we call **F**-value (standing for frequency-value) and is computed for function F as follows:

$$\mathbf{F}\text{-value} = \frac{\#\ \text{genes categorized in } F}{\#\ \text{genes}}.$$

We then show a more popular scoring measure: **P**-value between computed cluster C and standard function F:

$$\mathbf{P}\text{-value} = 1 - \sum_{i=0}^{N_{C,F}-1} \frac{\binom{N_F}{i}\binom{N - N_F}{N_C - i}}{\binom{N}{N_C}}.$$

This value measures the probability that C is enriched by F by chance. More concretely, if **P**-value is close to zero, the probability that a gene in the clusters

Table 1: Microarray expression data summary.

Name	#observations	#missing values	Reference
Brem	103	2172	[5]
Gasch	140	1462	[11]
Hughes	300	241	[18]
Spellman	57	2110	[33]
Storey	262	6946	[34]
Yvert	180	4405	[42]

is chosen by chance will be close to zero. That is, major functions of a cluster should have smaller **P**-values. The **P**-value has been well used in annotating tools, such as GOTermFinder [4]. However, we note that **P**-value, which originates from the hypergeometric distribution, heavily depends upon the number of standard functions. For example, even if all genes in a cluster have the same function, the corresponding **P**-value might not be small enough if the number of genes having this function is much larger than that of the cluster.

3. Experiments

3.1. *Data*

We focused on genes of *Saccharomyces Cerevisiae*. We used a list of sequence similarities which are already computed and stored in the SGD database [20]. This list has gene pairs with similarity p-values which are less than 0.01. We used six gene expression datasets shown in Table 1 to validate the performance of our clustering method. Missing values in these six data sets were all imputed by a method based on k-nearest neighbors [35] with $k = 10$.

We used GO (Gene Ontology) to generate a (gold-)standard dataset of fifteen gene functions, which was used through our experiments for some purposes, particularly for evaluating the clusters generated by our method. The manner of generating these standard functions is described in Appendix A. The GO was obtained from http://www.geneontology.org/ontology/gene_ontology.obo.

3.2. *Examining Gene Networks on Sequence Similarities*

We examined sequence similarities to generate a sequence network, following the manner of Step 1 of our method.

3.2.1. *Generating a Sequence Network*

Figure 4 shows the plots of Prec_v which were obtained by changing cut-off value v against p-values, where we used the fifteen standard functions generated in Section 3.1. From the figure, we can see that genes with high similarities in sequences

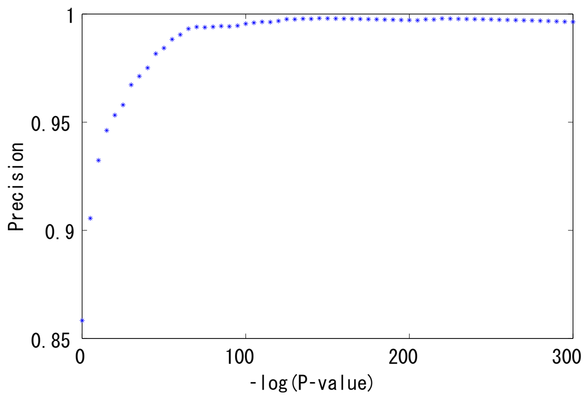

Fig. 4: Prec_v against sequence similarity p-values.

Table 2: Gene sets (in which a node is a gene) used in Sections 3.3 to 3.5.

Sets	# nodes	# edges	# nodes in LCC	# edges in LCC	Section used
I	2,025	7,972	487	4,450	3.3 & 3.4
II	2,305	9,710	582	5,686	3.4
III	2,903	12,642	834	8,229	3.5

are very likely to be in the same functional clusters (or share the same functions). From the figure, we chose the cut-off of 1.0e-5, corresponding to Prec_v of 0.9056, since the network at the next point by Prec_v of 0.9324 had only 90 nodes in the largest connected component (LCC) while the network at the cut-off of 1.0e-5 had 487 nodes in the LCC. We here note that a sequence network is generated from sequence similarities, and the fifteen standard functions were just used to decide the cut-off value which can give a network with some moderate size. Even if we use another standard set of functions, the cut-off value will be 1.0e-5 due to the network size of LCC, implying that the same sequence network will be generated, regardless of given standard functions.

At this cut-off value of 1.0e-5, we generated three different types of networks, i.e. gene sets: 1) Set I: was genes which can be categorized into one or more of the fifteen standard clusters, 2) Set II: was genes which can be labeled by any GO terms including the fifteen standard clusters and 3) Set III: was all genes including unknown genes or gene unlabeled with any GO terms. Table 2 shows the statistics of these three sets. Note that Set I $\subset$ Set II $\subset$ Set III. We used the LCC only throughout this paper.

Table 3: Normalized mutual information (R:Ratio cut, N: Normalized cut).

Dataset	ω_0	ω_1	ω^*	p-value (ω^* vs. ω_1)	R	N
(a) K=10						
Brem	0.0676	0.2694	**0.2949**	1.16e-13	0.2806	0.2799
Gasch	0.0595	0.2733	**0.2892**	4.86e-09		
Hughes	0.0549	0.2641	**0.2862**	7.10e-14		
Spellman	0.0417	0.2628	**0.3011**	3.55e-23		
Storey	0.0586	0.2715	**0.2955**	6.97e-13		
Yvert	0.0602	0.2662	**0.3019**	3.63e-21		
(b) K=15						
Brem	0.0766	0.3107	**0.3245**	9.90e-14	0.3185	0.3132
Gasch	0.0839	0.3104	**0.3263**	6.37e-16		
Hughes	0.0819	0.3090	**0.3267**	1.45e-15		
Spellman	0.0642	0.3117	**0.3309**	3.26e-16		
Storey	0.0953	0.3109	**0.3283**	1.96e-15		
Yvert	0.0866	0.3097	**0.3252**	4.63e-13		
(c) K=20						
Brem	0.0968	0.3493	**0.3679**	2.74e-17	0.3517	0.3418
Gasch	0.0990	0.3518	**0.3664**	2.92e-16		
Hughes	0.0974	0.3536	**0.3639**	1.23e-11		
Spellman	0.0746	0.3504	**0.3661**	6.75e-13		
Storey	0.1161	0.3539	**0.3665**	1.52e-17		
Yvert	0.1096	0.3497	**0.3660**	8.83e-16		

3.3. *Evaluating the Performance of Integrative Clustering Part*

We examined the performance of Step 2 of our method using microarray data and a sequence network, which was generated in the preceding section. We note that Step 3 of our method was not used in this experiment.

3.3.1. *Competing Methods*

We examined the performance of our clustering method (Step 2) with changing ω from zero (ω_0) to one (ω_1) at the interval of 0.1. Out of the eleven results obtained in this manner, we chose the optimal ω^* which gave the minimum cost. We compared the clustering performance at ω^* with those of ω_0 and ω_1. Furthermore, we compared the performance of our clustering method with two graph clustering methods using the sequence network only, i.e. graph clustering with ratio cut [8, 15] and normalized cut [28].

3.3.2. *Data and Procedure*

We used Set I and six gene expression datasets in Table 1. As described in Section 2.3, our clustering algorithm uses k-means at the final step, and the results by

k-means depend on initial cluster centers. Thus we performed 100 trials with different random initial centers for each ω and averaged over the 100 runs to make a fair comparison.

3.3.3. *Evaluation Criterion*

We checked the performance of our method by comparing resultant clusters with gold-standard clusters, for which we used fifteen GO functions we generated in Section 3.1. For evaluation criterion, we used normalized mutual information (NMI) which is defined in information theory as a quantity to measure the amount of information shared between two random variables. NMI is defined by the following formula:

$$\mathrm{NMI} = \frac{I(V, V')}{\sqrt{H(V) \cdot H(V')}} \tag{8}$$

where V and V' are the calculated clusters and standard clusters, respectively. $I(V, V')$ is the mutual information between V and V', and $H(V)$ and $H(V')$ are the entropy of V and V'. This can be estimated by the following formula [43]:

$$\mathrm{NMI} = \frac{\sum_{C,C'} N_{C,C'} \log(\frac{N \cdot N_{C,C'}}{N_C N_{C'}})}{\sqrt{(\sum_C N_C \log \frac{N_C}{N})(\sum_{C'} N_{C'} \log \frac{N_{C'}}{N})}} \tag{9}$$

where C is a computed cluster and C' is a standard function. NMI takes a value ranging from 0 to 1, and the closer to one it is, the more similar to standard clusters the computed clusters are. We note that NMI is the most typical measure for checking the performance of a clustering method.

3.3.4. *Performance Comparison*

Table 3 shows the averaged NMI for the six datasets at $K = 10$, 15 and 20. For all cases of this table, the averaged NMI at ω^* was slightly better than that at ω_1, which was clearly better than that at ω_0. Since the difference ω^* and ω_1 looked small, we then applied paired t-test between them to confirm the statistical significance in the difference between them, and p-values of this test are also reported in the fifth column. Table 3 further shows the averaged NMI obtained by using ratio cut and normalized cut, which were almost the same level as those by ω_1. All these results imply that the optimal ω outperformed ω_0 and ω_1, meaning that combining the two different datasets by our method was significantly effective for gene clustering. As ω^* outperformed others for all K, we used K=10 in Sections 3.4 and 3.5.

To further evaluate the effectiveness of the cost function, we then focused on Spellman, which was with the highest NMI and the smallest p-value in Table 3 (a), and checked the consistency between the NMI and the cost function. Figure 5 shows the costs (in the top row) and NMI (in the bottom row) with varying ω from zero to one for the three cases of K: 10, 15 and 20. Blue points in each figure show 100

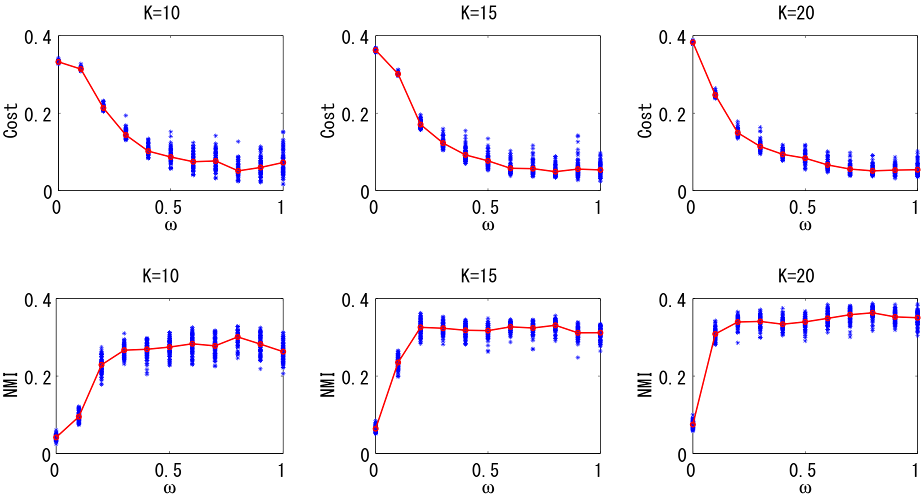

Fig. 5: Top: total costs of K=10, 15 and 20; Bottom: NMI of K=10, 15 and 20.

values corresponding to 100 repeated runs at each trial, and a red curve shows the average over the 100 runs. One can see that for all three cases of K, the ω with the minimum cost resulted in the highest NMI, meaning that our strategy of using the cost function to find $\omega*$ was useful. Another finding was that the optimal ω in the cost was 0.8 for all three cases, and then hereafter we set ω^* at 0.8.

3.4. *Evaluating the Predictive Performance of Our Entire Method*

We examined the performance of our method including Step 3. Here let us make clear the difference between Sections 3.3 and 3.4. In Section 3.3, genes were clustered and this grouping was evaluated by comparing with standard functions (clusters). On the other hand, in Section 3.4, as will be described later, cross-validation was performed, meaning that genes were divided into training and testing, and once clustered, functions of each gene in the test set were predicted by the functions of training genes in the cluster into which the test gene was fallen. Thus one can say that clustering performance was checked in Section 3.3, while predictive performance was examined in Section 3.4.

3.4.1. *Competing Methods*

We first examined three parameter settings of our method: ω^*, ω_0 and ω_1 with **F**-value and **P**-value. We then considered a simple approach, Random, with the following procedure: We first assumed ten clusters and, for each cluster, randomly assigned a GO term with a score randomly taking a value between zero and one. We then randomly assigned one of ten cluster labels to each gene as a predicted cluster label.

Next we examined the performance of network-based approaches. In fact, ω_1 of our method is equal to a network-based clustering of normalized network modularity cut over a sequence network. We, from gene expressions, first generated a network, which we call an *expression network*, by drawing an edge between two genes if the cosine similarity in expressions of the two genes is larger than some prefixed cut-off value θ. We then performed clustering nodes of the following three types of networks: 1) an expression network only (ExpNet), 2) the intersection of an expression network and a sequence network (IntSecNet) and 3) the union of the two networks (UnionNet). Clustering over these three networks was done by spectral clustering with normalized network modularity to make a fair comparison with our method. We tested both **F**-value and **P**-value.

We further examined the performance of "joint profile"-based approaches. We can first generate a feature vector from each gene sequence by using its amino acid compositions, hydrophobicity, normalized van der Waars volume, polarity and polarizability, which are well used in this type of transformation, e.g. [7]. We then examined three types of feature vectors as inputs for k-means: 1) expression profile only (ExpKM), which is basically equal to ω_0, 2) sequence feature only (SeqKM) and 3) their combination (ComKM). We further explored the performance of "joint profile"-based approaches by examining SVM, fixing the cluster set at the fifteen standard clusters, which were generated in Section 3.1. As well, this case, we used the above three networks, and we call the results obtained by 1), 2) and 3), ExpSVM, SeqSVM and ComSVM, respectively. For ExpSVM, we used the cosine similarity kernel to make a fair comparison with our method, and for SeqSVM, we tested three different kernels: linear (SeqLiSVM), polynomial (SeqPoSVM) and Gaussian (SeqGaSVM). The kernel of ComSVM was the simple addition of the cosine similarity kernel of ExpSVM and SeqGaSVM, which was chosen for SeqSVM since in our experiments there were no significant difference in performance among three SeqSVMs, comparing to ω^*. SVM is a binary classifier, and so for fifteen standard functions, we generated fifteen SVMs by repeating training a SVM using genes of each function as positives and others as negatives. Prediction was performed by taking the highest prediction score among the predictions of all fifteen trained SVMs. For SVM, we used Matlab with the QP (Quadratic Programming) option and optimized the regularization parameter by examining a wide variety of possible values.

3.4.2. *Data and Procedure*

We conducted a 100x5-fold cross validation over Set II. That is, we divided all genes into five blocks of roughly equal size, and at each trial four out of five were used for training and the remaining one was for test. We repeated this division 100 times and the results were averaged over the 100 runs. More precisely, in all clustering approaches, each run is that all genes were first clustered, and functions of each gene in the test set were predicted by using only genes in the training dataset. On the other hand, in SVM-based approaches, training and test genes are separately used for training and testing phases, respectively.

We used Set II for training and Set I for testing. However, for SVM-based approaches, we used Set I for training, since in classification, we needed a fixed size of true classes. This implies that a larger number of genes are available in training of clustering than that of classification. The Spellman dataset was used as gene expressions.

3.4.3. *Evaluation Criterion*

We used Set I genes for testing, meaning that all genes in testing can be categorized into one or more fifteen standard functions that we generated in Section 3.1. We note that except SVM, we used a larger number of GO clusters in training.

We evaluated all competing methods in this experiment in a binary classification manner by regarding each pair of a gene and a function as an example. First, for each pair of gene i and function k, we can assign one (positive) if gene i has function k; otherwise zero (negative). This can be a true label of the pair of gene i and function k. Second, a clustering method in this experiment can output a score whether gene i has function k, because $\mathbf{F}$-value (or a similar quantitative score) of function k can be computed from the cluster into which gene i was fallen. (We note that similarly SVM-based approaches can also easily output a score for each pair of gene i and function k, because for each function, an SVM was trained.) We then sorted examples (pairs) according to resultant scores and computed AUC (Area Under the ROC (Receiver Operator Characteristic) curve) by using the sum of the rank of negatives. For the detailed procedure of computing AUC in binary classification task, interested readers can consult some work on AUC, e.g. [16, 19].

3.4.4. *Results*

Table 4 shows the AUC of all compared methods. First, the AUC of ω^* was significantly better than those of ω_1 and ω_0, being consistent with Table 3. Naturally the AUC of Random was almost 0.5, being equal to random guessing. For network-based approaches, cut-off θ for cosine similarities is a parameter to change the number of edges in an expression network. In fact, expression networks had 87,538 edges at θ of 0.25 and 13,826 edges at θ of 0.50 while only 1,016 edges at 0.75. This difference changed the AUC of IntSecNet, which was around 0.69 at 0.25 and 0.50 but dropped drastically at 0.75. This is probably because the intersection network at θ of 0.75 was too sparse. However, in ExpNet, the performance was relatively stable through the three values of θ. In UnionNet, AUC clearly raised with increasing θ, implying that this combination worked in prediction. In fact, AUC at θ of 0.75 reached 0.7789 which looked very close to that of ω^*. We then computed paired t-test to examine the significance between these two methods and found that its p-value was 1.0358e-61, implying that the difference in AUC was statistically significant. All results of "joint-profile"-based approaches were much

Table 4: AUC of competing methods.

Method	AUC		
Our method			
ω_0	0.6728		
ω_1	0.7359		
ω^*	**0.7828**		
Simple approach			
Rand	0.4792		
Network-based approaches			
	θ=0.25	θ=0.50	θ=0.75
ExpNet	0.6547	0.6491	0.6502
IntSecNet	0.6934	0.6910	0.6368
UnionNet	0.7051	0.7443	0.7789
k-means-based approaches			
	k=10	k=15	k=20
ExpKM	0.6594	0.6530	0.6619
SeqKM	0.6832	0.6954	0.6837
ComKM	0.6988	0.6856	0.6997
SVM-based approaches			
SeqLiSVM	0.6629		
SeqPoSVM	0.6766		
SeqGaSVM	0.6709		
ExpSVM	0.6724		
ComSVM	0.6827		

lower than AUC of ω^*. In particular, the results of SeqSVM were worse than ω_1 through all three kernels, while ExpSVM was comparable to ω_0, implying that for sequences, a network-based approach, especially UnionNet was more favorable than using sequence features. Another item of note is that there might be some room to improve the performance of SVM-based methods, but the results in Table 4 imply that our method will outperform them even if some improvements are done on the SVM-based methods.

Table 5 shows the AUC of the competing methods when **P**-value was used in prediction. The AUC of each method in this table was much worse than that of the corresponding method of Table 4, indicating that **F**-value was more useful than **P**-value.

Overall, we can conclude that ω^* with **F**-value was the best among competing methods for predicting gene functions, including network-based, k-means-based and SVM-based approaches.

Table 5: AUC of competing methods with **P**-value.

Method	AUC		
Our method			
ω_0	0.5839		
ω_1	0.6951		
ω^*	0.7270		
Network-based approaches			
	θ=0.25	θ=0.50	θ=0.75
ExpNet	0.5405	0.5062	0.5463
IntSecNet	0.5161	0.6153	0.5323
UnionNet	0.6347	0.6907	0.7246
k-means-based approaches			
	k=10	k=15	k=20
ExpKM	0.5644	0.5425	0.5605
SeqKM	0.6123	0.6164	0.6287
ComKM	0.6334	0.6089	0.6369

3.5. *Annotating Yeast Genome*

As we confirmed the performance advantage of our method over a lot of other methods, we applied our method to annotate functions of genes whose functions are still unknown.

3.5.1. *Data and Procedure*

We used Set III and Spellman. We run ω^* with k of 10, 1,000 times, and then predicted gene functions by using the average clustering results over the 1,000 runs and **F**-value.

3.5.2. *Results*

Table 6 shows resultant ten clusters, with uncharacterized genes and GO terms with **F**-values of more than 0.5. Some uncharacterized genes in SGD are already annotated by another database. We validated unknown genes in Table 6 by checking FunCat [26] of CYGD [14]. In particular, we focused on the top three clusters in Table 6 show the analysis on them. In C_1, YLR289w, the only uncharacterized gene in C_1, was annotated to "translation" in FunCat, being the same as the GO term which was ranked at the top by **F**-value in Table 6. In C_2, we found that all genes were annotated by FunCat to either "Modification by phosphorylation, dephosphorylation, autophosphorylation" or "Phosphate metabolism", except only two genes YGL083w and YJL057c, which were uncharacterized in FunCat. These two functional categories were mostly consistent with the predicted function: "Protein amino acid phosphorylation", which was GO:6468. In fact, "Modification by phosphorylation, dephosphorylation, autophosphorylation" and "Phosphate metabolism" were GO:16310 and GO:6796, respectively, and GO:6468 was the only child of GO:16310, which was also the only child of GO:6796. This fact indicates that these functions were firmly related with each other. In C_3, all genes were annotated by FunCat to "tRNA processing" (except two genes: YGL131c and YPR022c), which was a descendant of "transcription", while the predicted function "Regulation of transcription from RNA polymerase II promoter" was also a descendant of "transcription" in GO, meaning that these two functions were also related with each other. Overall these observations imply the preciseness in annotation by our approach.

4. Conclusion and Discussion

We have proposed a new method for predicting gene functions using sequence similarities and microarray expressions. From our thorough experiments, we have shown that our method can optimally combine two different types of datasets, sequences and expressions, for clustering genes and precisely predict functions of unknown genes using the clusters obtained. The effectiveness of our method was validated by its advantage over a lot of other methods in performance comparison.

The datasets for genes we used in our experiments were sequences and microarray expressions, both of which can be measured for any gene basically. We note that currently available biological data can be divided into roughly two classes: 1) relatively static data that include gene/protein sequences and phylogenetic profiles, and 2) biologically active (dynamic) data which are obtained by experimental observations like gene expressions and protein-protein interactions. A favorable aspect on the choice of gene sequences and microarray expressions is that we selected one dataset from each of the above two classes. In this sense, we can say that our method generated gene clusters by balancing between static and dynamic information on functional activities of genes.

We used the Spellman gene expression dataset [33] in our experiments. In out method, it is easy to replace the Spellman dataset with another dataset of gene expressions. If we choose another type of expression dataset, we will be able to include another type of dynamic information on genes in the combination of two data sources. Thus interesting future work would be to use some specific experimental conditions of gene expressions to annotate gene functions in the corresponding cell condition. We further note that our framework allows to use any type of existing functional categories of genes such as FunCat [26] and Pfam [10] as standard gene functions, instead of GO which we used in our experiments. Thus we can choose any of them if it is more reliable and covers a lot of genes.

A biological dataset can be categorized into either of structured or unstructured data. Thus our method is able to use any addition as well as the current two datasets, meaning that we can combine more than two different types of datasets, although the addition might be not necessarily applied to any gene.

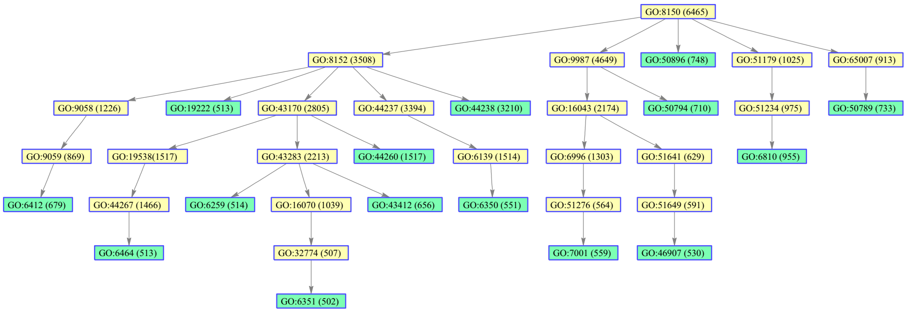

Fig. 6: Fifteen clusters obtained from GO.

In Section 3.5, we fix the size of clusters at ten, since in this setting the analysis by using FunCat on the clustering results was relatively easy. However, the GO information of ten clusters might be too vague to hypothesize the functions of unknown genes. If we use a larger number of clusters, we might be able to predict more detailed functions of unknown genes. We emphasize that in our framework, changing this parameter (K) is easy, and in fact we already performed the experiments when K is fifteen and twenty. Function prediction results under $K = 15$ and 20 are, of course, available upon request.

We used the data of *Saccharomyces Cerevisiae* only in our experiments. Obviously our annotation method is not specialized to this species but a general method, which can be applied to any species if we can make a gene network of sequence similarities from the sequence information of this species and have a microarray dataset on this species. Thus it would be also interesting to annotate genes of a more unfamiliar species with a lot of uncharacterized genes if for this species, there are datasets from which we can have enough information. From a different viewpoint, interesting future work might be to predict functions of genes of more than one species by using information on sequences and expressions of a wide variety of species at once.

Acknowledgements

The authors would like to thank Shanfeng Zhu, Ichigaku Takigawa and Raymond Wan for their helpful comments and fruitful discussion over this work. M. S. was supported in part by Kyoto University 21st Century COE Program "Knowledge Information Infrastructure for Genome Science", Grant-in-Aid for Young Scientists (B) by MEXT, Japan and BIRD of Japan Science and Technology Agency (JST). W-K. C. is supported in part by HKRGC Grant No. 7017/07P, HKUCRGC Grants, HKU Strategy Research Theme fund on Computational Sciences, Hung Hing Ying

Physical Research Sciences Research Grant, National Natural Science Foundation of China Grant No. 10971075 and Guangdong Provincial Natural Science Grant No. 9151063101000021. H. M. has been supported in part by BIRD of Japan Science and Technology Agency (JST).

Appendix A. Generating Standard Fifteen Clusters from Gene Ontology (GO)

The Gene Ontology (GO) consists of controlled vocabularies describing three aspects of gene product functions: (i) molecular function, (ii) biological process and (iii) cellular component. Each of these three aspects is called an ontology and represented by a directed acyclic graph (DAG) where each GO term is attached to a node in the graph. The root (or the starting point) exists in a DAG, and GO terms are arranged hierarchically from general ones to specific ones in a DAG. This means that if a gene is assigned to one term (or a node), then the gene can be assigned to all ancestors of this term. Thus the number of genes assigned to a node is not less than that to any of its children.

To build a sequence network, we generate standard clusters using GO in the following manner: We first place a cut-off value, which is the minimal cluster size. We then start from the root to its descendants, and if the number of genes attached to one node is less than this cut-off value, we go back to its parent and stop there so that the number of genes assigned to this node is more than the cut-off value. By repeating this procedure, we have a set of clusters, keeping the minimum cluster size being larger than the specified cut-off value. We note that we can change the minimum cluster size of the cluster set by changing the cut-off value.

In our experiments, we used the cut-off value of 500 by which all genes in GO were divided into fifteen clusters. Figure 6 shows a tree with the fifteen clusters corresponding to the leaves (colored green) of this tree, and the GO term (accession ID) and the number of genes at each node are shown for all nodes in this tree. The ontology for each GO accession ID in Fig. 6 is described in Table A1.

References

[1] Azuaje, F., Clustering-based approaches to discovering and visualising microarray data patterns. *Briefings in Bioinformatics*, 4(1):31–42, 2003.

[2] Barabási, A-L., Reka., A., Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] Basu, S., Bilenko, M., Mooney, R. J., A probabilistic framework for semi-supervised clustering. *Proc. Tenth ACM SIGKDD*, 59–68, 2004.

[4] Boyle, E. I., *et al.*, 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(18):3710–3715, 2004.

[5] Brem, R. B., *et al.*, Genetic dissection of transcriptional regulation in budding yeast. *Science*, 296:752–755, 2002.

[6] Brown, M., *et al.*, Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci.*, 97:262–267, 2000.

[7] Cai, C. Z., *et al.*, SVM-prot: web-based support vector machine software for functional classification of a protein from its primary sequence. *Nucl. Acids Res.*, 31(13):3692–3697, 2003.

[8] Chan, P. K., Schlag, M. D. F., Zien, J. Y., Spectral K-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, 1994.

[9] Ding, C., *et al.*, A min-max cut algorithm for graph partitioning and data clustering. *Proc. IEEE International Conference on Data Mining*, 107–114, 2001.

[10] Finn, R. D., *et al.*, Pfam: clans, web tools and services. *Nucleic. Acids Res.*, 34(Database issue):D247–51, 2006.

[11] Gasch, A. P., *et al.*, Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell*, 11(122):4241–4257, 2000.

[12] Guimera, R., Nunes Amaral, L. A., Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–9, 2005.

[13] Guimera, R., Sales-Pardo, M., Amaral, L. A. N., Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E*, 70:025101, 2004.

[14] Guldener, U., *et al.*, CYGD: the comprehensive yeast genome database. *Nucl. Acids Res.*, 33(Database issue):D364–D368, 2005.

[15] Hagen, L., Kahng, A. B., New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computed Aided Design of Integrated Circuits and Systems*, 11:1074–1085, 1992.

[16] Hand, D. J., Till, R. J., A simple generalization of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

[17] Hastie, T., Tibshirani, R., Friedman, J. H., *The Elements of Statistical Learning.* Springer, 2003.

[18] Hughes, T. R., *et al.*, Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, 2000.

[19] Mamitsuka, H., Selecting features in microarray classification using ROC curves. *Pattern Recognition*, 39(12):2393–2404, 2006.

[20] Nash, R., *et al.*, Expanded protein information at sgd: new pages and proteome browser. *Nucl. Acids Res.*, 35(Database issue):D468–D471, 2007.

[21] Newman, M. E. J., Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004.

[22] Newman, M. E. J., Girvan, M., Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

[23] Ng, A. Y., Jordan, M. I., Weiss, Y., On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14*, 849–856. 2002.

[24] Ravasz, E., *et al.*, Hierarchical organization of modularity in metabolic networks. *Science*, 297(5589):1551–1555, 2002.

[25] Rost, B., Enzyme function less conserved than anticipated. *J. Mol. Biol.*, 318:595–608, 2000.

[26] Ruepp, A., The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucl. Acids Res.*, 32:5539–5545, 2004.

[27] Sharan, R., Ulitsky, I., Shamir, R., Network-based prediction of protein function. *Molecular Systems Biology*, 3(88):1–13, 2007.

[28] Shi, J., Malik, J., Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[29] Shiga, M., Takigawa, I, Mamitsuka, H., Annotating gene function by combining expression data with a modular gene network. *Bioinformatics*, 23(13):i459–i467, 2007.

[30] Shiga, M., Takigawa, I., Mamitsuka, H., A spectral clustering approach to optimally

combining numerical vectors with a modular network. *Proc. Thirteenth ACM SIGKDD*, 647–656, 2007.

[31] Smith T. F., Waterman, M. S., Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[32] Song, C., Havlin, S., Makse, H. A., Self-similarity of complex networks. *Nature*, 433:392–395, 2005.

[33] Spellman, P. T., *et al.*, Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 92(12):3273–3297, 1998.

[34] Storey, J. D., Akey, J. M., Kruglyak, L. Multiple locus linkage analysis of genomewide expression in yeast. *PLoS Biol.*, 3:e267, 2005.

[35] Troyanskaya, O., *et al.*, Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[36] von Luxburg, U., A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[37] Wagstaff, K., Cardie, C., Clustering with instance-level constraints. *Proc. Seventeenth International Conference on Machine Learning*, 1103–1110, 2000.

[38] Wagstaff, K., *et al.*, Constrained k-means clustering with background knowledge. *Proc. Eighteenth International Conference on Machine Learning*, 577–584, 2001.

[39] Watts, D. J., Strogatz, S. H., Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[40] White, S., Smyth, P., A spectral clustering approach to finding communities in graphs. *Proc. Fifth SIAM International Conference on Data Mining*, 76–84, 2005.

[41] Wu, L. F., *et al.*, Large-scale prediction of Saccharomyces cerevisiae gene function using overlapping transcriptional clusters. *Nature Genetics*, 31(3):255–265, 2002.

[42] Yvert, G., *et al.*, Transacting regulatory variation in saccharomyces cerevisiae and the role of transcription factors. *Nature Genetics*, 35(1):57–64, 2003.

[43] Zhong, S., Ghosh, J., A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:11–1037, 2003.

Table 6: Annotation for 151 uncharacterized genes in yeast genome

C	Uncharacterized genes	GO terms	**F**-value
C_1	YLR289w	GO:6412	0.8000
		GO:43283	0.7333
		GO:16043	0.5333
C_2	YBR028c YDL025c YGL083w YGR052w YJL057c YKL161c YKL168c YMR291w YPL141c YPL150w YPL236c YPR106w	GO:6468	0.7255
C_3	YDL048c YER130c YGL131c YGR067c YLR375w YML081w YPL230w YPR013c YPR015c YPR022c	GO:6357	0.5152
		GO:16043	0.5152
C_4	YBL066c YBR033w YBR150c YBR239c YDR520c YER184c YFL052w YFR057w YIL130w YJL016w YJL206c YKL222c YKR064w YLL054c YLR187w YLR278c YMR019w YNR063w YPR196w	GO:44237	0.5439
		GO:43283	0.5088
C_5	YER047c YFR023w YGR250c YJR061w YPL074w	GO:43170	0.6212
		GO:44238	0.6212
		GO:44237	0.5758
C_6	YFR038w YGR130c YGR196c YJR134c YKL050c YKL105c YKR016w YBR148w YCR051w YDL070w YDR026c YEL043w YER033c YFR016c YLR247c YMR031c YMR086w YPL009c YPL186c YPL2602 YPL263C	GO:6996	0.5225
		GO:44237	0.5225
		GO:44238	0.5112
C_7	YAR020c YBL111c YBL112c YBL113c YBR301w YCR104w YDR291w YDR332w YDR542w YEL049w YEL075c YEL076c YEL077c YER150w YER189w YFL020c YFL064c YFL065c YFL066c YGL261c YGR294w YHL046c YHL049c YHL050c YHR218w YHR219w YIL011w YIL176c YIL177c YJL223c YJL225c YKL224c YLL025w YLL064c YLL066c YLL067c YLR037c YLR461w YLR462w YLR464w YMR244w YNR076w YOL161c YOR394w YPL282w	GO:6139	0.6792
		GO:43283	0.6792
		GO:3996	0.6226
C_8	YBR108w YBR125c YFR022w YGR068c YIR003w YJL084c YKR021w YML118w YNL018c YNL034w YOL042w	GO:16043	0.6275
		GO:44238	0.6078
		GO:43283	0.5686
C_9	YAL065c YAR061w YAR062w YDL037c YDL211c YDR134c YFL051c YHR213w YHR214w YIL169c YJL078c YJL079c YJR151c YKR013w YMR317w YNL176c YOL036w YOR009w YOR227w	GO:16043	0.6216
C_{10}	YDR128w YDR267c YER066w YKL121w YMR102c YNL035c YPL183c YPL247c	GO:6996	0.6852
		GO:43283	0.6667
		GO:6139	0.5370

Table A1: Accession ID and Corresponding Ontology.

GO accession ID	Ontology
6139	nucleobase, nucleoside, nucleotide and nucleic acid metabolism
6259	DNA metabolism
6350	transcription
6412	protein biosynthesis
6464	protein modification
6351	transcription, DNA-dependent
6810	transport
6996	organelle organization and biogenesis
7001	chromosome organization and biogenesis (sensu Eukaryota)
8150	biological process
8152	metabolism
9058	biosynthesis
9059	macromolecule biosynthesis
9987	cellular process
16043	cell organization and biogenesis
16070	RNA metabolism
19222	regulation of metabolism
19538	protein metabolism
32774	RNA biosynthesis
43170	macromolecule metabolism
44237	cellular metabolism
43283	biopolymer metabolism
43412	biopolymer modification
44238	primary metabolism
44260	cellular macromolecule metabolism
44267	cellular protein metabolism
46907	intracellular transport
50784	cocaine catabolism
50794	regulation of cellular process
50896	response to stimulus
51179	localization
51234	establishment of localization
51276	chromosome organization and biogenesis
51641	cellular localization
51649	establishment of cellular localization
65007	biological regulation