A System to Find Genetic Networks Using Weighted Network Model

Tomohiro Moriyama [1]
moriyama@i.kyushu-u.ac.jp
Ayumi Shinohara [1]
ayumi@i.kyushu-u.ac.jp
Masayuki Takeda [1]
takeda@i.kyushu-u.ac.jp
Osamu Maruyama [2]
maruyama@ims.u-tokyo.ac.jp
Takao Goto[2]
takao@ims.u-tokyo.ac.jp
Satoru Miyano [2]
miyano@ims.u-tokyo.ac.jp
Satoru Kuhara [3]
kuhara@grt.kyushu-u.ac.jp

[1] Department of Informatics, Kyushu University 33, 6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
[2] Human Genome Center, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan
[3] Graduate School of Genetic Resources Technology, Kyushu University, 6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan

Abstract

We are developing a system which finds a genetic network from data obtained by multiple gene disruptions and overexpressions. We deal with a genetic network as a weighted graph, where each weight represents the strength of activation from a gene to another gene. In this paper, we explain the overview of our system, and our strategy to visualize the weighted network. We also study the computational complexity related to the visualization.

1 Introduction

Analyzing the interactions between genes by systematic gene disruptions and gene overexpressions is getting more important in Genome Science. DNA microarray technology [6, 14] enabled us to produce time series of gene expression patterns. Our research group have launched a project whose purpose is to reveal the gene regulatory networks among the 6,200 genes of *Saccharomyces cerevisiae* while many laboratories have also started similar project.

Some methods have been proposed to identify the gene regulatory networks from gene expression patterns [1, 4, 11, 13]. In our previous work [9, 10], we have introduced a *weighted network model* as an edge-weighted graph, where each weight reflects the strength of the interaction. We analyzed its computational complexity [9]. The simulation results showed that our algorithm to adjust weights incrementally predicts more accurate than the algorithm whose worst case performance is theoretically guaranteed [10].

In this paper, we explain the overview of our system. The core module of the system is to produce a genetic network as a weight matrix from given gene expression profiles. We have implemented our incremental weight adjusting algorithm, whose practical behaviors have verified in our previous work [10]. The problem here is that the size of the produced network is quite large. If we directly apply our system for gene expression profiles consisting of 6,200 genes, the output is a weight matrix of size $6,200 \times 6,200$. Unfortunately, there is no guarantee that the network is sparse. Thus it will be quite hard for the users to understand the connections among genes in the network. The standard graph drawing techniques [2] will not work effectively.

We introduce two modules in our system in order to reduce the problem. The first one is the correlation analyzer of gene activations. It analyzes not only the standard correlation coefficient of

gene activations, but also the correlation coefficient through some functions specified by the users. We can use it to condense the gene expression profile data, with respect to the users' interests.

The second module is to visualize the weighted network. We propose two views to capture the connections in the network. One is the global view, which rearranges a weight matrix by permuting columns and rows so that similar genes locate closely and dissimilar genes far from each other. The goal is related to other methods of visualization, such as the Self-Organizing Map (SOM) [7] and the Multi-Dimensional Scaling (MDS) [8]. We analyze the computational complexity to rearrange the weight matrix optimally. We show that the problem can be reduced to the traveling salesman problem and vice versa. These results imply that the problem is computationally intractable to optimize, while some heuristic methods may be applicable.

The other view is called the local view, which shows how strongly a specified gene activates other genes, and how strongly it is activated by other genes on the Cartesian coordinates. Through the local view, the users can observe the behavior of each gene separately.

The rest of the paper is organized as follows. In Section 2, we review the weighted network model. Section 3 explains the modified weighted majority algorithm, which makes the weighted network from the gene expression profile data. Section 4 shows the method for selecting correlating genes. In Section 5, we consider how to visualize the network effectively, and investigate the computational complexity of the problems related to the visualization.

2 Weighted Network Model

We first explain the format of a gene expression profile data, which will be the input of our system (Fig. 1). An example of the profile is shown in Fig. 2, where the number of genes is $n = 6153$ and the number of experiments is m. Each experiment corresponds to three rows in the table. The first row represents the strength of the green (control), and the second row represents that of red. We denote by *green*(i, j) (*red*(i, j), resp.) the strength of green (red, resp.) value of gene g_i at j-th experiment. The third row represents the disruption or overexpression. We denote *flag*$(i, j) = 1$ if gene g_i at jth experiment is either disrupted or overexpressed, *flag*$(i, j) = 0$ otherwise.

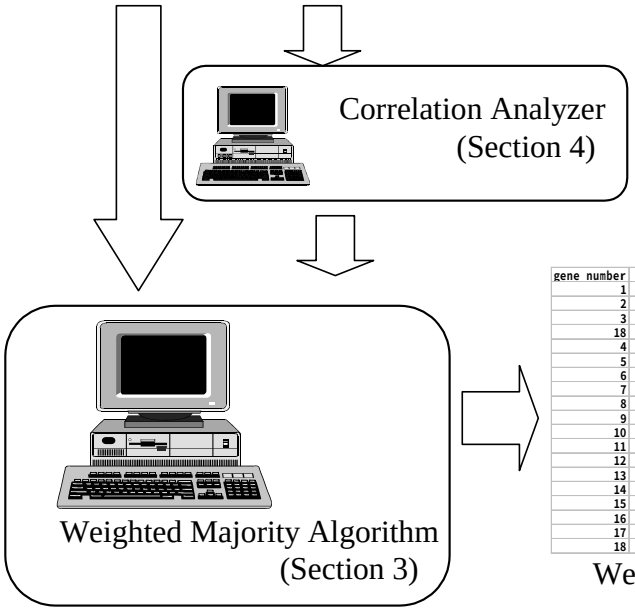
We now briefly review our genetic network model by following [9]. We denote by $\mathcal{R}$ the set of all real numbers. A *genetic network* $G = (V, E, W, T)$ is a weighted directed graph with the set V of nodes (genes), the set $E \subseteq V \times V$ of directed edges, the weight mapping $W : E \to \mathcal{R}$, and the threshold mapping $T : V \to \mathcal{R}$. We assume that the graph has no self-loop, i.e, $(g_i, g_i) \notin E$ for any $g_i \in V$, although the graph may contain cycles.

For each edge $e = (g_i, g_k) \in E$, the value $W(e)$ represents the strength of *activation* from g_i to g_k. Note that the value $W(e)$ may be negative, which means g_i *inactivates* g_k. The value $T(g_i)$ expresses the *threshold* associated with a gene g_i, explained later. Each gene is assumed to be in an *active* state or an *inactive* state. For convenience, we denote $a(g_i) = 1$ if a gene g_i is active, $a(g_i) = -1$ if inactive. A *gene regulation rule* assigned to a gene g_i is expressed by the balance between the weighted sum $sum(g_i) = \sum_{(g_k,g_i)\in E} a(g_k) \cdot W((g_k, g_i))$ and the threshold value $T(g_i)$. If $sum(g_i) > T(g_i)$, the gene g_i should be active state, if $sum(g_i) < T(g_i)$, the gene g_i should be inactive state, We do not care if the sum is equal to the threshold.

For example, consider the gene network in Fig. 3, where $T(g_5) = 0$. The gene g_5 should be active if g_1 and g_3 are active and g_4 is inactive, since the weighted sum $1 \cdot 3 + 1 \cdot 4 + (-1) \cdot (-2) = 9 > 0 = T(g_5)$. On the other hand, g_5 should be inactive if g_1 is active and g_3 and g_4 is inactive, since the weighted sum $1 \cdot 3 + (-1) \cdot 4 + 1 \cdot (-2) = -5 < 0 = T(g_5)$. The state of g_5 is independent of the state of g_2, since there is no edge from g_2 to g_5. Hereafter, we represent the weight mapping W as an $n \times n$ weight matrix $W = (w_{ik})$ where $n = |V|$, so that each element w_{ik} expresses the strength of activation from g_i to g_k.

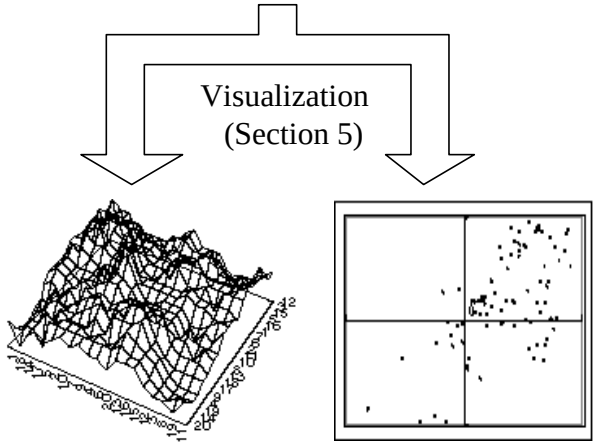
		experment 1			experment 2				experment m		
gene name	gen	green	red	flag	green	red	flag		green	red	flag
YHR007C ERG11	1	6825	11920	0	6028	6741	0		7703	8380	0
YBR218C PYC2	2	5238	6837	0	9361	11070	0		3897	3795	0
YAL051W FUN43	3	4353	3448	0	3443	3338	0		3291	4006	0
YAL053W	4	2895	5743	0	4655	5323	0		3769	3199	0
YAL054C ACS1	5	948	479	0	733	460	0		415	358	0
YAL055W	6	3102	2836	0	1170	794	0		786	628	0
YAL056W	7	2066	1716	0	2240	2262	0		1781	1725	0
YAL058W CNE1	8	2500	2997	0	1530	1571	0		835	682	0
YOL109W	9	5080	6516	0	6523	10071	0		7811	10544	0
YCR084c TUP1	703	3535	339	1	4490	5507	0		6509	5837	0
YPR016C	6153	6044	8048	0	7244	6842	0		3775	2711	0

Input :gene express profile data



gene number	49	50	51	52	53	54
1	0	0	0	0	0	0.981862 x
2	0	0	0	0	0	0
3	0	0	0	0	0	0
18	0	0	0	0.973458 l(	0	0.973098 l(
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0.973458 l(	0	0	0	0
9	0	0	0	0	0.979855 l(	0
10	0	0	0	0	0	0
11	0	0	0.973458 l(	0	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	0	0	0	0	0	0
15	0	0	0	0	0	0
16	0	0	0	0	0	0
17	0	0	0	0	0	0
18	0	0	0	0.973458 l(	0	0.973098 l(

Weighted Matrix (Section 2)



Global View

Local View

Figure 1: Overview of the system.

		1st. exp.			2nd. exp.			mth. exp.		
gene name		green	red	flag	green	red	flag	green	red	flag
YHR007C ERG11	1	6825	11920	0	6028	6741	0	7703	8380	0
YBR218C PYC2	2	5238	6837	0	9361	11070	0	3897	3795	0
YAL051W FUN43	3	4353	3448	0	3443	3338	0	3291	4006	0
YAL053W	4	2895	5743	0	4655	5323	0	3769	3199	0
YAL054C ACS1	5	948	479	0	733	460	0	415	358	0
YAL055W	6	3102	2836	0	1170	794	0	786	628	0
YAL056W	7	2066	1716	0	2240	2262	0	1781	1725	0
YAL058W CNE1	8	2500	2997	0	1530	1571	0	835	682	0
YOL109W	9	5080	6516	0	6523	10071	0	7811	10544	0
YCR084c TUP1	703	3535	339	1	4490	5507	0	6509	5837	0
YPR016C	6153	6044	8048	0	7244	6842	0	3775	2711	0

Figure 2: Input gene expression profile data.

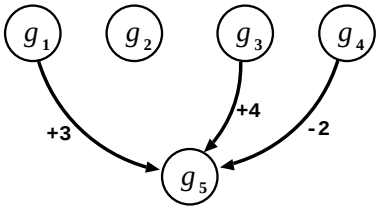


Figure 3: Gene regulatory rule, where $T(g_5) = 0$.

3 Finding Genetic Network from Gene Profile Data

This section explains the main module of our system. The input is a gene expression profile data, and the output is a weight matrix W of a gene network.

The method described in this section consists of two stages. The first stage is to make a $n \times m$ intermediate matrix $A = (a_{ij})$ whose values are ternary value $+1, 0$ or -1 from the input gene expression profile data. Each a_{ij} represents state of gene i at jth experiment, $a_{ij} = +1$ if active, $a_{ij} = -1$ if inactive, and $a_{ij} = 0$ if unknown (neutral). The second is the core module of weighted majority algorithm, which constructs a genetic network from intermediate matrix A and $B = (b_{ij})$, where $b_{ij} = flag(i, j)$.

3.1 Preprocessing

We first explain our basic idea to discretize the profile data consisting of $red(i, j)$ and $green(i, j)$ into ternary values $\{1, 0, -1\}$ representing the state of the gene. We denote the state of gene g_i at the j-th experiment by a_{ij}. Ideally, the state of the gene should be either active or inactive. However, because of the noise in the experiments, it is not always easy to judge the state of genes from data. Thus we introduce a neutral state, or unknown state, which means that we regard the gene is neither active nor inactive. We denote the neutral state by $a_{ij} = 0$.

Basically we judge the state of each gene by the ratio $ratio(i, j)$ of red value to green value, that is $ratio(i, j) = red(i, j)/green(i, j)$. When the ratio is drastically changed through the m experiments, we determine a_{ij} separately for each experiment, based on the value $ratio(i, j)$ relative to the m experiments. If the ratio is relatively high, we regard that the gene is active at the experiment. If the ratio is relatively low, the gene is inactive at the experiment. On the other hand, when the ratio

does not changes so much through the m experiments, it is not reasonable to assign distinct values to each a_{ij} according to the small differences of $ratio(i,j)$. Therefore we assign $a_{i1} = a_{i2} = \cdots = a_{im} = 1$ when all $red(i,j)$ are high, and we assign $a_{i1} = a_{i2} = \cdots = a_{im} = -1$ when all $red(i,j)$ are low. Since the threshold values are sensitive to the gene network output by the system, we designed the system so that the users can specify the five control parameters $x_1, x_2, \ldots, x_5$. Typical values of these parameters are $x_1 = 2$, $x_2 = 0.7$, $x_3 = 0.3$, $x_4 = 10,000$, and $x_5 = 2,000$. Details are described as follows.

We use the log-transformed values $r_{\log}(i,j)$ of the ratios $ratio(i,j)$, that is, $r_{\log}(i,j) = \log_2 ratio(i,j)$. For each $i = 1, \ldots, n$, let

$$\begin{array}{rcl} r^{\min}_{\log}(i) & = & \min_{1 \le j \le m}\{r_{\log}(i,j)\}, \quad \text{and} \\ r^{\max}_{\log}(i) & = & \max_{1 \le j \le m}\{r_{\log}(i,j)\}. \end{array}$$

If $r^{\max}_{\log}(i) - r^{\min}_{\log}(i) \ge x_1$, we set

$$a_{ij} = \begin{cases} 1 & \text{if } x_2 \le r'(i,j), \\ 0 & \text{if } x_3 < r'(i,j) < x_2, \\ -1 & \text{if } r'(i,j) \le x_3, \end{cases}$$

where

$$r'(i,j) = \frac{r_{\log}(i,j) - r^{\min}_{\log}(i)}{r^{\max}_{\log}(i) - r^{\min}_{\log}(i)}.$$

Otherwise, that is, if $r^{\max}_{\log}(i) - r^{\min}_{\log}(i) < x_1$, we set

$$a_{ij} = \begin{cases} 1 & \text{if } r''(i,j) \ge x_4 \text{ for all } j = 1, \ldots, m, \\ -1 & \text{if } r''(i,j) \le x_5 \text{ for all } j = 1, \ldots, m, \\ 0 & \text{otherwise,} \end{cases}$$

where

$$r''(i,j) = ratio(i,j) \cdot \frac{1}{m} \sum_{1 \le j \le m} green(i,j).$$

Remark that $r''(i,j)$ is the value $ratio(i,j)$ multiplied by the average of green values over m experiments.

3.2 Modified Weighted Majority Algorithm

In Fig. 4, we show the heuristic algorithm which we have implemented as a core module. The input is a pair of an $n \times m$ intermediate matrix A over $\{-1, 0, 1\}$ and an $n \times m$ matrix $B = (b_{ij})$ over $\{0, 1\}$. The value b_{ij} is 1 if and only if the gene g_i is either disrupted or overexpressed at the j-th experiment. The output is a weight matrix $W = (w_{ik})$ of a genetic network. In order to deal with ternary states $\{-1, 0, 1\}$, we divide the threshold value $T(g)$ of gene g into the *positive threshold* $T_+(g)$ and the *negative threshold* $T_-(g)$. In this implement, for every gene g, we fixed both the positive threshold $T_+(g) = \theta^+ = (x_2 - x_3) \cdot n/2$ and $T_-(g) = \theta^- = -(x_2 - x_3) \cdot n/2$, where x_2 and x_3 are given by the users in the preprocessing phase.

In this algorithm, the matrix W is represented by the difference $W^+ - W^-$ of two matrices. Each element in W^+ and W^- takes a value between 0 and 1. Initially, all elements in W^+, W^- are set to 0. If the weights are not fit to given observed data, they will be adjusted as follows: if the weight was too small, we increase it by setting $w^+ := (w^+_{kj} + 1)/2$ and $w^- := w^-/2$. If the weight was too large, we decrease it by setting $w^+ := w^+/2$ and $w^- := (w^- + 1)/2$. We note that the update strategy is slightly modified in the prototype of the system described in [9]. The adjusting process is repeated until either no more adjustment is required or the loop counter becomes greater then pre-defined threshold.

Algorithm Weights Adjusting
Given A : $n \times m$ matrix of values $\{-1, 0, 1\}$ and
B : $n \times m$ matrix of values $\{0, 1\}$
begin
initialize W^+ and W^- to be the zero matrices;
for each $i \in \{1, \ldots, n\}$ in random order **begin** /* *using Multi-threads* */
loop-counter := 0 ;
repeat
for $j := 1$ **to** m **begin** /* *each experiment*/
if $b_{ij} = 0$ **then begin** /* *neither disrupted nor overexpressed* */
Let $sum := \sum_{k=1}^{n} a_{ki}(w_{ki}^+ - w_{ki}^-)$;
if ($a_{ij} = 1$ **and** $sum < \theta^+$) **or** ($a_{ij} = 0$ **and** $sum < \theta^-$) **then**
for $k := 1$ **to** n **do**
if $a_{kj} = 1$ **then**
increase(k, i)
else if $a_{kj} = -1$ **then**
decrease(k, i)
else if ($a_{ij} = -1$ **and** $sum > \theta^-$) **or** ($a_{ij} = 0$ **and** $sum > \theta^+$) **then**
for $k := 1$ **to** n **do**
if $a_{kj} = 1$ **then**
decrease(k, i)
else if $a_{kj} = -1$ **then**
increase(k, i)
end
loop-counter := loop-counter + 1;
until W^+ and W^- become stable **or** loop-counter > limit-of-loop;
end;
return the matrix $W = W^+ - W^-$. **end.**

procedure increase(k, i: **integer**) /* *increase* w_{ki} */
begin
$w_{ki}^+ := (w_{ki}^+ + 1)/2$;
$w_{ki}^- := w_{ki}^-/2$
end

procedure decrease(k, i: **integer**) /* *decrease* w_{ki} */
begin
$w_{ki}^+ := w_{ki}^+/2$;
$w_{ki}^- := (w_{ki}^- + 1)/2$;
end

Figure 4: Weights Adjusting Algorithm.

4 Correlation Analyzer of Genes

In this section, we describe a submodule of the system which investigates correlation coefficients between genes in gene expression profile data. The input is a gene expression profile data and the output is a matrix whose element at i-th column and k-th row is the correlation coefficient between the gene g_i and g_k, which is explained below. As an option, the users can specify a subset of genes to be investigated. Moreover, when the users specify a single gene g_i and a threshold parameter $l \in [0, 1]$, the system outputs a clipped gene expression profile data consisting of all genes whose absolute values of correlation coefficients to g_i are greater than l. In this way, the submodule can be used to *condense* the gene expression profile data with respect to the users' interests.

Not only the standard correlation coefficient, the submodule can compute a correlation coefficient with respect to specified functions. Let $\mathcal{F}$ be a set of functions from $\mathcal{R}$ to $\mathcal{R}$, and $f \in \mathcal{F}$. For short, we denote $r_f(i, j) = f(r''(i, j))$, where r'' is defined in the previous section. The average of $r_f(i, j)$ for a gene g_i is $\bar{r}_f(i) = \frac{1}{m}\sum_{j=1}^{m} r_f(i, j)$. Let f and $h \in \mathcal{F}$. The *correlation coefficient between the genes g_i and g_k with respect to f and h*, denoted by $C_{i,k}(f, h)$, is defined as

$$C_{i,k}(f,h) = \frac{\sum_{j=1}^{m}\{(r_f(i,j) - \bar{r}_f(i))(r_h(k,j) - \bar{r}_h(k))\}}{\sqrt{\sum_{j=1}^{m}(r_f(i,j) - \bar{r}_f(i))^2}\sqrt{\sum_{j=1}^{m}(r_h(k,j) - \bar{r}_h(k))^2}}.$$

The pool $\mathcal{F}$ of functions can be determined by the users. As candidates, currently, three functions x^2, $\log x$ and $\log^2 x$ are available. For each pair of genes g_i and g_k, the submodule exhaustively searches for the pair of functions f and h in $\mathcal{F}$ which maximizes the absolute value of $C_{i,k}(f, h)$. This procedure is applied to all pair of genes.

5 Visualization

This section argues how to visualize the weighted graphs which will be output by our system.

If the graph is relatively small and sparse, some existing techniques for graph-drawing [2] might be helpful. However, if the graphs which we have to deal with are quite large whose size is about $6,200$, and possibly dense, it is difficult for researchers to understand the interaction among the genes in the network. Here we propose two methods to visualize the networks: *global view* and *local view*. The former intends to capture the similarity of the genes in the network, whereas the latter is to figure out the behavior of individual gene.

5.1 Global view

We consider how to give a global view to a weighted graph which is large and dense. A well known method is to apply the Multi-Dimensional Scaling (MDS) [8]. MDS is an iterative algorithm for visually analyzing the structure of high-dimensional data. MDS produces a non-linear mapping of data which preserves interpoint distances of high-dimensional data while reducing to a lower dimensionality, say, two or three dimension. We can apply MDS in order to arrange genes so that similar genes are close to each other and dissimilar genes far from each other. Another method is the Self-Organizing Map (SOM) [7] which is useful to visualize and interpret large high-dimensional data. SOM can also applicable to arrange genes. Although these two heuristic methods are often used, (for example, [3]), their performances are not analyzed very well to our best knowledge. Here we formulate the task to arrange genes as optimization problems, and show their computational complexity.

Intuitively, our approach can be illustrated as Fig. 5. The figures are generated from a weight matrix $W = (w_{ik})$ by plotting the strength of activation/inactivation from gene g_i to gene g_k at position (i, k) as a height, with activation being high and inactivation being low. The right figure is derived from the left figure by permuting rows and columns of the matrix so that the surface becomes

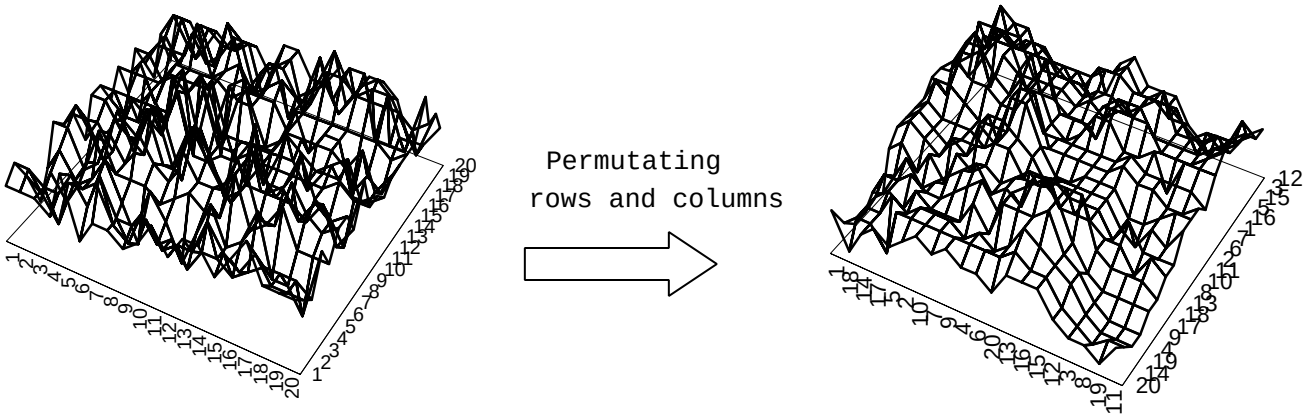


Figure 5: Two dimensional permutation of a weight matrix.

smooth. The smoothness of the surface implies that similar genes tend to gather close each other on each axis. From this matrix, we can get the interaction among genes. Thus the problem is to minimize the total difference of the values with their neighbors. First we consider the difference with the four neighbors.

Definition 5.1. *Two-dimensional Permutation Problem* **(2dPerm)** is, given an $n \times m$ matrix A, to find a matrix B which is generated by permuting rows and columns of A, so that B minimizes the following cost function:

$$cost(B) = \sum_{1 \leq i \leq n, 1 \leq j \leq n} (|b_{i+1,j} - b_{i,j}| + |b_{i-1,j} - b_{i,j}| + |b_{i,j+1} - b_{i,j}| + |b_{i,j-1} - b_{i,j}|).$$

We also formulate the one dimensional version of the problem, where we consider only two neighbors.

Definition 5.2. *One-dimensional Permutation Problem* **(1dPerm)** is, given an $n \times m$ matrix A, to find a matrix B which is generated by permuting rows and columns of A, so that B minimizes the following cost function:

$$cost(B) = \sum_{1 \leq i \leq n, 1 \leq j \leq m} (|b_{i+1,j} - b_{i,j}|).$$

We can show the complexities of these problems as follows.

Theorem 5.3. Both **1dPerm** and **2dPerm** are **NP**-hard. Moreover, both problems can be approximately solved in polynomial-time within the approximation ratio 1.5.

Proof (Overview) We can show the NP-hardness of these problems as follows. If each element of the matrix in **1dPerm** is either 0 or 1, **1dPerm** can be regarded as the Traveling Salesman Problem in Hamming Space, which is known to NP-hard [12]. Moreover, **1dPerm** is reducible to **2dPerm**, which implies that **2dPerm** is also NP-hard.

On the other hand, we can show an approximation-preserve reductions from **2dPerm** to **1dPerm**, and from **1dPerm** to the Metric Traveling Salesman Problem, for which a polynomial-time approximation algorithm with approximation ratio 1.5 is known [5] □

By the above theorem, we see that these problems are quite hard, although some heuristic methods are applicable. We note that the problem **1dPerm** is closely related to the SOM approach, although **2dPerm** does not directly related to SOM.

5.2 Local View

The local view of the system visualizes in a two dimensional plain that how a specific gene activates other genes and how the gene is activated by the other genes (Fig. 6).

The system plots each gene g_i $(i \neq k)$ at the position whose x-coordinate is w_{ik} and y-coordinate is w_{ki}. That means, if the gene g_k is strongly activated by the gene g_i, it locates at the right-side, and if the gene g_k is strongly inactivated by the gene g_i, it locates at the left-side. On the other hand, if the gene g_k strongly activates the gene g_i, its location is on the upper side, and if it strongly inactivates g_i, the location is on the lower side. Therefore genes whose activities are independent from the gene g_i gather in the center origin. In this way, we can capture the activity of each gene intuitively. Moreover, we can also classify the genes into a number of categories with respect to the specified gene g_i, by clustering the points in the plain.

In the left figure in Fig. 6, we see that the specfied gene is activated from many other genes. On the other hand, from the right figure, we realize that the specified gene activates and inactivates many genes. In this way, we can capture the behavior of each gene separately.

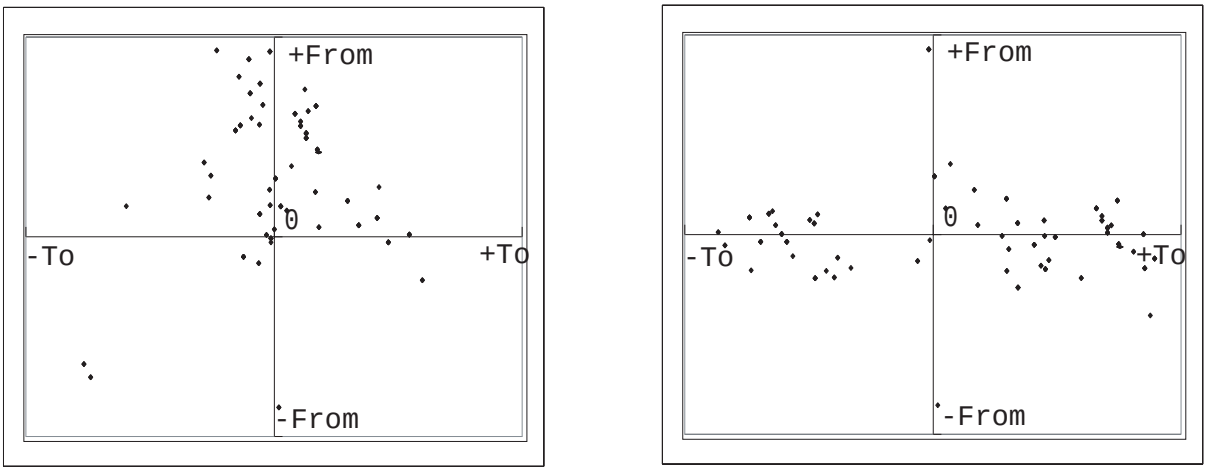


Figure 6: Local views.

6 Conclusion

We explained our system to find a genetic network from gene expression profile data using weighted network model. We have finished implementing the core module and the correlation analyzer on our workstations, using multi-threads. The response of the system seems to be quite reasonable. As soon as the experimental data are available, we will report our results on genetic networks produced by the system. We are now implementing the visualization modules.

References

[1] Akutsu, T., Miyano, S., and Kuhara, S., Identification of genetic networks from a small number of gene expression patterns under the boolean network model, *Pacific Symposium on Biocomputing '99*, 17–28, 1999.

[2] Battista, G., Eades, P., Tamassia, R., and Tollis, I.G., *Graph Drawing —algorithms for the visualization of graphs*, Prentice Hall, 1999.

[3] http://cmgm.stanford.edu/pbrown/sporulation/additional/websom.html.

[4] Chen, T., He, H., and Church, G., Modeling gene expression with differential equations, *Pacific Symposium on Biocomputing '99*, 29–40, 1999.

[5] Christofides, N., Worst-case analysis of a new heuristic for the traveling salesman problem, *Proc. Symposium on New Directions and Recent Results in Algorithms and Complexity*, page 441, Academic Press, 1976.

[6] DeRisi, J., Lyer, V., and Brown, P., Exploring the metabolic and genetic control of gene expression on a genomic scale, *Science*, 278:680-686, 1997.

[7] Kohonen, T., *Self-Organization and Associative Memory*, Vol. 8, Springer Series in Information Sciences, 1984.

[8] Kruskal, J., *Multidimensional scaling and other methods for discovering structure*, John Wiley & Sons, 1977.

[9] Noda, K., Shinohara, A., Takeda, M., Matsumoto, S., Miyano, S., and Kuhara, S., Finding genetic network from experiments by weighted network model, *Genome Informatics 1998*, 141–150, 1998.

[10] Noda, K., Shinohara, A., Takeda, M., Matsumoto, S., Miyano, S., and Kuhara, S., Simulation results on finding genetic networks by weighted network model, *Pacific Symposium on Biocomputing (Poster abstracts)*, page 84, 1999.

[11] Szallasi, Z., Genetic network analysis in light of massively parallel biological data acquisition, *Pacific Symposium on Biocomputing '99*, 5–16, 1999.

[12] Trevisan, L., When Hamming meets Euclid: The approximability of geometric TSP and Steiner tree, In *Proc. 29th ACM Symposium on Theory of Computing*, 21–29, 1997.

[13] Weaver, D., Workman, C., and Stormo, G., Modeling regulatory networks with weight matrices, In *Pacific Symposium on Biocomputing '99*, 112–123, 1999.

[14] Yuh, C.-H., Bolouri, H., and Davidson, E., Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene, *Science*, 279:1896–1902, 1998.