Finding Genetic Network from Experiments by Weighted Network Model

Kiyoshi Noda [1] knoda@i.kyushu-u.ac.jp
Ayumi Shinohara [1] ayumi@i.kyushu-u.ac.jp
Masayuki Takeda [1] takeda@i.kyushu-u.ac.jp
Satoshi Matsumoto [2] matumoto@ss.u-tokai.ac.jp
Satoru Miyano [3] miyano@hgc.ims.u-tokyo.ac.jp
Satoru Kuhara [4] kuhara@grt.kyushu-u.ac.jp

[1] Department of Informatics, Kyushu University 33, Fukuoka 812-8581, Japan.
[2] Faculty of Science, Tokai University, Kanagawa 259-1291, Japan.
[3] Human Genome Center, University of Tokyo, Tokyo 108-8639, Japan.
[4] Graduate School of Genetic Resources Technology, Kyushu University, Fukuoka 812-8581, Japan.

Abstract

We study the problem of finding a genetic network from data obtained by multiple gene disruptions and overexpressions. We define a genetic network as a weighted graph, and analyze the computational complexity of the problem. We show that if there exists a weighted network which is consistent with given data, we can find it in polynomial time. Moreover, we also consider the optimization problem, where we try to find an optimally consistent weighted network with given data. We show that the problem is NP-hard. On the other hand, we give a polynomial-time approximation algorithm to solve it with approximation ratio 2. We report some simulation results on experiments.

1 Introduction

One of the hottest research topic in Genome Science is to analyze the interactions between genes by systematic gene disruptions and gene overexpressions. Our research group has installed a systematic experimental method which allows both multiple gene disruptions and gene over expressions. By using this method, we have launched a project whose purpose is to reveal the gene regulatory networks between the 6,200 genes of *Saccharomyces cerevisiae* while many laboratories have also started similar project. Among many computational problems related to the project, in this paper, we focus on the problem of finding a genetic network from limited experimental data.

Akutsu *et al.* [1] defined a genetic network as a boolean network, where each node is either in an active state or an inactive state. In their model, each edge represents an interaction of two genes, either activation or inactivation. In this paper, we introduce a *weighted network model* as an edge-weighted graph, where each weight reflects the strength of the interaction. We formulate the problem as a decision problem, and analyze its computational complexity. We show that if there exists a weighted network which is consistent with given data, we can find it in polynomial time. However, the assumption is impractical, since experimental data may contain errors. Moreover, if a network has cycles, observed data themselves may not be consistent with each other. Thus we have to consider the optimization problem, where we try to find an optimally consistent weighted network with given data. We show that the problem is NP-hard, so that we have little hope to solve it efficiently. On the other hand, we also show a polynomial-time approximation algorithm to solve it with approximation ratio 2. We also propose an algorithm to adjust weights incrementally, inspired by the weighted majority strategy [5, 8].

From a practical point of view, we are interested in the behavior of these two algorithms to predict unknown state, after learning some limited given data. As training data, we use a normal state and all states observed by flipping the activity of each gene one by one. This is because our actual experiments for Saccharomyces cerevisiae will start from such trials. In order to estimate the prediction ability, we use a set of states observed by flipping activities of two genes at a time. We report these simulation results, and compare these two algorithms.

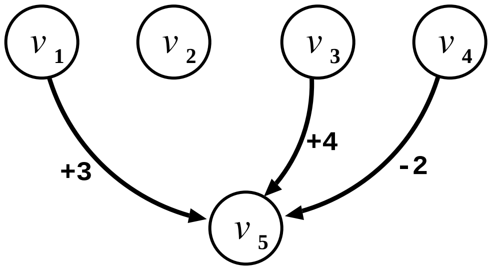

Figure 1: Gene regulatory rule, where $T(v_5) = 0$.

2 Weighted Network Model

We define a genetic network by partially following [1]. 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, 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, $(v, v) \notin E$ for any $v \in V$, although the graph may contain cycles. We call a node of G a *gene*. For each edge $e = (v_i, v_j) \in E$, the value $W(e)$ represents the strength of *activation* from v_i to v_j. Note that the value $W(e)$ may be negative, which means v_i *inactivates* v_j. The value $T(v)$ expresses the *threshold* associated with a gene v, explained later.

Each gene is assumed to be either in an *active* state or an *inactive* state. For convenience, we denote $a(v) = 1$ if a gene v is active and $a(v) = -1$ if inactive.

A *gene regulation rule* assigned to a gene v is expressed by the balance between the weighted sum $\sum_{(v',v)\in E} a(v') \cdot W((v', v))$ and the threshold value $T(v)$. If the sum is greater than the threshold, the gene v should be active, and if the sum is less than the threshold, v should be inactive. We do not care if the sum is equal to the threshold.

For example, the gene v_5 in Fig. 1 should be active if v_1 is active and v_3 and v_4 are inactive, since the weighted sum $1\cdot 3+(-1)\cdot 4+(-1)\cdot(-2) = 1$ is greater than the threshold value $T(v_5) = 0$. On the other hand, v_5 should be inactive if v_1 is active and v_3 and v_4 is inactive, since $1\cdot 3+(-1)\cdot 4+1\cdot(-2) = -5 < 0$. The state of v_5 is independent of the state of v_2, since there is no edge from v_2 to v_5.

For a gene v, *gene disruption* of v enforces v inactive and *gene overexpression* of v enforces v active. Let $x_1, \ldots, x_p$ and $y_1, \ldots, y_q$ be mutually distinct genes of G. An *experiment* with gene overexpressions of $x_1, \ldots, x_p$ and gene disruptions of $y_1, \ldots, y_q$ is denoted by $\langle x_1, \ldots, x_p, \neg y_1, \ldots, \neg y_q\rangle$.

A *global state* of G is a mapping $\psi : V \to \{-1, 1\}$. Each global state must satisfy $\psi(x_i) = 1$ and $\psi(y_i) = -1$ under an experiment $\langle x_1, \ldots, x_p, \neg y_1, \ldots, \neg y_q\rangle$, although the global state need not be consistent with the gene regulation rules. We say that a global state ψ of G is *stable* under an experiment $\langle x_1, \ldots, x_p, \neg y_1, \ldots, \neg y_q\rangle$ if it is consistent with all gene regulation rules except those assigned to the genes in $\{x_1, \ldots, x_p, y_1, \ldots, y_q\}$. That is, for each node $v \notin \{x_1, \ldots, x_p, y_1, \ldots, y_q\}$,

$$\psi(v) = \begin{cases} 1 & \text{if } \sum_{(v',v)\in E} \psi(v') \cdot W((v', v)) > T(v), \\ -1 & \text{if } \sum_{(v',v)\in E} \psi(v') \cdot W((v', v)) < T(v). \end{cases}$$

Otherwise, it is called *unstable*. We say that a genetic network G is *stable* under an experiment α if there is a global state of G which is stable under α. When no experiment is made on G, we simply remove "under α" from the terminology.

We note that the stable global state is not necessarily unique. For example, for a simple network G_1 in Fig. 2, both of the global states $(v_1, v_2) = (1, 1), (-1, -1)$ are stable. On the other hand, for some networks, there may be no global state which is stable. The network G_2 in Fig. 2 is a simple instance of such ones. Moreover, a stable network may become unstable under some experiments. We introduce a *cost* of global state s with respect to a network G, as the number of gene regulation rules in G which is not satisfied by s. Thus, the cost of a stable global state is 0. A *minimum cost global state* under

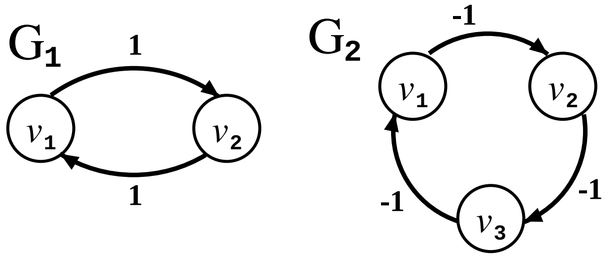

Figure 2: two simple genetic networks, where all threshold values are 0.

an experiment α is an arbitrary global state whose cost is minimum among all possible global states. Therefore, all stable global states are minimum cost global states. For the network G_2 in Fig. 2, all of the global states $(v_1, v_2, v_3) = (-1,-1,1)$, $(-1,1,-1)$, $(1,-1,-1)$, $(1,1,-1)$, $(1,-1,1)$, $(-1,-1,1)$ are minimum cost global states, since all of their costs are one, and no stable global state exists for G_2. We assume that all observed global states produced by experiments are minimum cost states.

Remark 2.1. Theoretically, the class of functions which we introduced is the class of linear separable functions. Since it contains both AND functions and OR functions stated in [1] as a subclass, we think that the class is rich enough. Moreover, we may express each inactive state v by $a(v) = 0$ instead of $a(v) = -1$, since these two models are equivalent.

This paper investigates the possibility to construct a genetic network from a given set of observed global states, which are produced by experiments. We are interested in the complexity of the problem, and its practical behavior.

3 Matrix representation of Genetic Network

In order to treat the problem more formally and to analyze the computational complexity, we reformulate the problem by representing a genetic network as a matrix. For a positive integer n, we denote by $\mathcal{N}(n)$ the set of integers $\{1, 2, \ldots, n\}$. We denote by $M(m, n; U)$ the set of all $m \times n$ matrices over a domain U. For a matrix $A \in M(m, n; U)$, a_{ij} denotes the element at column i and row j for each $i \in \mathcal{N}(m)$ and $j \in \mathcal{N}(n)$.

We denote a genetic network $G = (V, E, W, T)$ with $V = \{v_1, v_2, \ldots, v_n\}$ by a pair of matrices $\tilde{W} \in M(n, n; \mathcal{R})$ and $\tilde{T} \in M(1, n; \mathcal{R})$. Each value w_{ij} in the matrix $\tilde{W}$ corresponds to the weight $W((v_i, v_j))$, and $w_{ij} = 0$ if $(v_i, v_j) \notin E$. Since G does not contain self-loop, each diagonal element w_{ii} is always zero. The matrix $\tilde{T}$ directly corresponds to the threshold mapping T. In the sequel, we will confuse the weight mapping W with its matrix representation $\tilde{W}$, and the threshold mapping T with $\tilde{T}$.

A global state for G can be represented as a vector $\vec{a} = (a_1, \ldots, a_n) \in M(1, n; \{-1, 1\})$, where a_j denotes the state of gene v_j. $a_j = 1$ if the gene v_j is active, and $a_j = -1$ if inactive. Then the vector $\vec{s} = (s_1, \ldots, s_n)$ whose j-th element s_j represents the weighted sum $\sum_{(v', v_j) \in E} a(v') \cdot W((v', v_j))$ of the gene v_j can be computed by the matrix multiplication:

$$\begin{aligned}\vec{s} = (s_1, \ldots, s_j, \ldots, s_n) &= (\textstyle\sum_{i=1}^n a_i w_{i,1}, \ldots, \sum_{i=1}^n a_i w_{i,j}, \ldots, \sum_{i=1}^n a_i w_{i,n}) \\ &= (a_1, \ldots, a_n) \cdot \begin{pmatrix} w_{1,1} & \ldots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \ldots & w_{n,n} \end{pmatrix} = \vec{a} \cdot W\end{aligned}$$

Thus the *gene regulation rule* assigned to a gene v_j is expressed by the following: if $s_j > t_j$ then v_j should be active, and if $s_j < t_j$ then v_j should be inactive.

We express an experiment $\langle x_1, \dots, x_p, \neg y_1, \dots, \neg y_q \rangle$ by a vector $\vec{b} \in M(1, n, \{-1, 0, 1\})$, where

$$b_j = \begin{cases} 1 & \text{if } x_j, \\ -1 & \text{if } \neg y_j, \\ 0 & \text{otherwise.} \end{cases}$$

That is, $b_j = 1$ if the gene v_j is overexpressed, and $b_j = -1$ if v_j is disrupted. Otherwise, $b_j = 0$. In this way, we can express a pair of experiment and its result by two vectors $\vec{b}$ and $\vec{a}$. A series of l experiments and their results by a pair of matrices $B \in M(l, n; \{-1, 0, 1\})$ and $A \in M(l, n; \{-1, 1\})$ in an obvious way. We are now ready to formulate the problem to find a genetic network which is consistent with given observed global states as follows.

Definition 3.1. Finding stable genetic network problem

- Given $A \in M(l, n; \{-1, 1\})$ and $B \in M(l, n; \{-1, 0, 1\})$
- Find $W \in M(n, n; \mathcal{R})$ and $T \in M(1, n; \mathcal{R})$ such that
  (1) for each $j \in \mathcal{N}(n)$, $w_{jj} = 0$, and
  (2) for each $i \in \mathcal{N}(l)$ and $j \in \mathcal{N}(n)$, if $b_{ij} = 0$ then
  $s_{ij} > t_j \Leftrightarrow a_{ij} = 1$ and $s_{ij} < t_j \Leftrightarrow a_{ij} = -1$, where $S = A \cdot W$.

We note that the condition (1) is indispensable, since without it, a trivial solution exists; for example, W is the unit matrix and T is the zero vector.

Now we deal with the computational complexity of the problem. By reducing the problem into the linear programming problem, we can solve the problem in polynomial time.

Theorem 3.2. The finding stable genetic network problem can be solved in polynomial time.

By Theorem 3.2, the problem turned out to be tractable. Probably we can solve the problem of reasonably large size by applying some linear programming packages [3, 9]. However, in general, the above problem is too restricted in the sense that we only consider stable networks. As we stated in Fig. 2, some networks have no stable states, if they contain cycles. Moreover, in practice, due to the experimental noise, there is no guarantee to the existence of stable network. Therefore we have to deal with a more realistic problem, that is a natural extension of the problem.

By relaxing the second condition of the problem, we treat it as a maximizing problem.

Definition 3.3. Maximum satisfying genetic network problem (MaxSGN).

- Given: $A \in M(l, n; \{-1, 1\})$ and $B \in M(l, n; \{-1, 0, 1\})$.
- Find: $W \in M(n, n; \mathcal{R})$ and $T \in M(1, n; \mathcal{R})$ with $w_{jj} = 0$ for each $j \in N(n)$, which maximizes the cardinality of the set

$$\left\{ (i, j) \in \mathcal{N}(l) \times \mathcal{N}(n) \;\middle|\; \begin{array}{l} \text{if } b_{ij} = 0 \text{ then} \\ \quad s_{ij} > t_j \Leftrightarrow a_{ij} = 1 \text{ and} \\ \quad s_{ij} < t_j \Leftrightarrow a_{ij} = -1 \end{array} \right\},$$

  where $S = A \cdot W$.

At a first glance, the problem seems to be a quite complicated optimization problem. However, as we will show, the problem is very closely related to the following simple problem.

Algorithm MaxSGNtoMaxFLS
Given $A \in M(l, n; \{-1, 1\})$ and $B \in M(l, n; \{-1, 0, 1\})$
begin
  foreach $k = 1, \ldots, n$ **do**
    $S_k = \phi$
    foreach $i = 1, \ldots, l$ **do**
      if $b_{ik} = 0$ **then**
        if $a_{ik} = 1$ **then**
          $S_k := S_k \cup \{\text{“}\sum_{j=1}^{n} a_{ij} w_{jk} > t_k\text{”}\}$
        else   /* $a_{ik} = -1$ */
          $S_k := S_k \cup \{\text{“}\sum_{j=1}^{n} a_{ij} w_{jk} < t_k\text{”}\}$
  return $S_1, \ldots, S_n$
end

Figure 3: Converter from a instance of MaxSGN to n instances of MaxFLS$^{>}$

Definition 3.4 ([2]). Maximum feasible subset of linear inequalities problem (MaxFLS$^{>}$).

- Given: a set S of linear inequalities.
- Find: a solution which satisfies as many linear inequalities in S as possible.

The complexity of MaxFLS$^{>}$ is known as follows.

Theorem 3.5 ([2]). The maximum feasible subset of linear inequalities problem is NP-hard, even if each coefficient is either -1 or 1. Moreover, there exists a polynomial-time approximation algorithm for the problem with approximation ratio 2.

We can show the following theorem.

Theorem 3.6. The maximum satisfying genetic network problem is NP-hard. Moreover, there exists a polynomial-time approximation algorithm for the problem with approximation ratio 2.

Proof For NP-hardness, we can show that a log-space reduction from MaxFLS$^{>}$ with coefficients $\{-1, 1\}$ to MaxSGN.

Here, we briefly explain our polynomial-time approximation algorithm for MaxSGN whose approximation ratio is guaranteed to be 2. First we convert a given instance $A \in M(l, n; \{-1, 1\})$ and $B \in M(l, n; \{-1, 0, 1\})$ of MaxSGN into n instances $S_1, \ldots, S_n$ of MaxFLS$^{>}$ by the procedure in Fig. 3. Thus each S_k consists of at most l linear inequalities over variables $\{w_{1k}, \ldots, w_{nk}, t_k\}$. We assign $w_{kk} = 0$ for each k. Then we apply AK-algorithm in Fig. 4, proposed by Amaldi and Kann [2], in order to solve MaxFLS$^{>}$ for each S_k. Let $opt(S_k)$ be the cardinality of maximum feasible subset of S_k, and let $AK(S_k)$ be the number of inequalities in S_k which are satisfied by the output values $\{w_{1k}, \ldots, w_{nk}, t_k\}$ of AK-algorithm. Since $AK(S_k) \geq opt(S_k)/2$, we have $\sum_{k=1}^{n} AK(S_k) \geq \sum_{K=1}^{n} opt(S_k)/2$, which guarantees the approximation ratio of our algorithm. Since AK-algorithm runs in polynomial time with respect to the length of S_k, our algorithm also runs in polynomial time. □

The above theorem implies that if we try to find a genetic network which is optimally consistent with observed data, it is intractable. On the other hand, if we relax the condition of optimality, then the problem becomes tractable. Therefore it is important to develop practical algorithms using some heuristics methods in order to find a good genetic network. In the next section, we will propose a practical algorithm, inspired by the HAKKE system [5, 8]. We also note that our problem is very closely related to a classical problem of learning by a perceptron, where some learning algorithms are proposed [4, 6, 7].

Algorithm AK-algorithm
Given S: set of linear inequalities over variables X
begin
  while $S \neq \phi$ **do**
    if there are inequalities in S that contains a single variable **then**
      $U := \{x \in X \mid x$ occurs as a single variable in at least one inequalities of $S\}$;
      pick at random $y \in U$;
      $F(y) := \{e \in S \mid e$ contains only the variable $y\}$;
      assign a value to y that satisfies as many inequalities in $F(y)$ as possible ;
      $S := S - F(y)$;
    else
      pick at random $y \in X$;
      assign a random value to y ;
      reevaluate the inequalities in S that contain y;
      $X = X - \{y\}$;
  return the assignments
end.

Figure 4: AK algorithm for MaxFLS$^>$.

4 Weights Adjusting Algorithm

We propose an algorithm in Fig. 5 which outputs a genetic network as a pair of weighted matrix W and threshold vector T, from given observed data represented by matrices A and B. In this algorithm, the matrix W is represented by the sum $W^+ + W^-$ of two matrices $W^+ \in M(l, n; \mathcal{Z}^+)$ and $W^- \in M(l, n; \mathcal{Z}^-)$, where $\mathcal{Z}^+$ ($\mathcal{Z}^-$) denotes the set of non-negative (non-positive, resp.) integers. Similarly, the threshold vector T is also represented by two vectors T^+ and T^-. Initially, all elements in W^+, W^-, T^+, and T^- are set to 1, except that $w^+_{ii} = w^-_{ii} = 0$. If the weights and thresholds are not fit to given observed data, they will be adjusted by multiplying by two the corresponding elements in the matrices. The adjusting process is repeated until it converses or the loop counter becomes greater then pre-defined threshold.

5 Experimental Results

This section reports our simulation results on performing WA algorithm and AK algorithm. We estimated the learning ability of algorithms as follows. We randomly generate gene networks of size at most 1000 as target networks. The outdegree of networks varies from 1 to 8, although the indegree is not bounded. Target network may contain cycles. For each network, first we generated a global state *Normal*. Then we generated a set *Flip1* of global states by experiments of either a single overexpression or a single disruption for each gene. Thus the size of *Flip1* is equal to the number of genes. We input both *Normal* and *Flip1* to our algorithms as training examples. In order to verify the effect of learning, we randomly generated a set *Flip2* of 100 global states by experiments, where two genes are either overexpressed or disrupted. We counted the number of states in *Flip2* which are correctly predicted by using W and T output by the algorithms. We repeated this process for four times, and evaluated the average. The running time of AK algorithm was approximately one hour, while that of WA algorithm was approximately three hours for each trial, when the number of nodes was set to 1000.

First we investigated the relation between the outdegree and the accuracy. The outdegree of the target network was changed from 1 to 8, while the number of nodes was fixed to 1,000. The result is shown Table 1 and Fig. 6. As the outdegree increases, the accuracy goes down for both algorithms, which

Algorithm Weights Adjusting
Given $A \in M(l, n; \{-1, 1\}),\ B \in M(l, n; \{-1, 0, 1\})$
begin
  initialize W^+, W^-, T^+, T^-;
  for each $i \in \mathcal{N}(l)$ in random order **begin**
    $a_{i0} := 1$; loop-counter := 0 ;
    repeat
      for $j = 1$ **to** n **do**
        if $b_{ij} = 0$ **then**
          $sum := 0$;
          for $k = 0$ **to** n **do** $sum := sum + a_{ik}(w^+_{kj} - w^-_{kj})$;
          if $a_{ij} = 1$ and $sum \leq 0$ **then**
            for $k = 0$ **to** n **do**
              if $a_{ik}w^+_{kj} > 0$ **then** $w^+_{kj} := 2w^+_{kj}$
              else $w^-_{kj} := 2w^-_{kj}$;
          else if $a_{ij} = -1$ and $sum \geq 0$ **then**
            for $k = 0$ **to** n **do**
              if $a_{ik}w^+_{kj} < 0$ **then** $w^+_{kj} := 2w^+_{kj}$
              else $w^-_{kj} := 2w^-_{kj}$;
      loop-counter := loop-counter + 1;
    until W^+, W^-, T^+, T^- are stable **or** loop-counter > limit-of-loop;
  end;
end.

Figure 5: Weights Adjusting Algorithm

Table 1: Outdegree vs Accuracy

outdegree	1	2	4	6	8
WA	99.99%	93.48%	70.88%	63.14%	58.80%
AK	97.24%	86.47%	70.42%	69.51%	69.87%

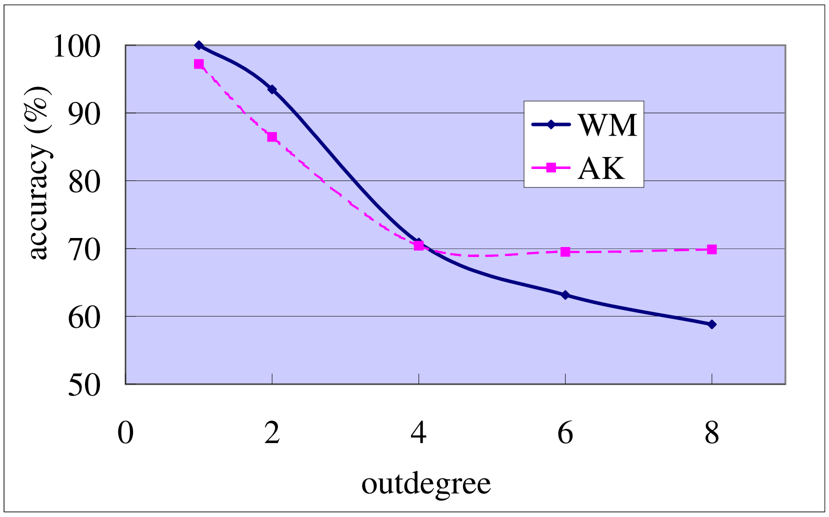

Figure 6: Outdegree vs Accuracy, where the number of nodes is 1000.

were naturally expected. WA is superior to AK when the outdegree is less than 4, and inferior otherwise.

Next, we examined the relation between the number of nodes and the accuracy. We changed the number of nodes from 10 to 1,000, while the outdegree was fixed either 2 or 4. The results are shown in Table 2, and illustrated in Fig. 7 and Fig. 8. In case that the outdegree is 2, the accuracies of both algorithms do not go down even when the number of nodes increases. However, in case that the outdegree is 4, the accuracy of WA goes down as the number of nodes increases.

6 Concluding Remarks

In this paper, we formulated a genetic network as a weighted graph, and we investigated the computational complexity of the problem. We have implemented two algorithms to find networks from given observed data. In our experiment, we dealt with only 1 (active) and -1 (inactivate) as observed data. However, our model can be easily generalized to treat degrees of activation if needed.

Table 2: Number of nodes vs Accuracy, where d denotes the outdegree.

number of nodes		10	32	100	316	1,000
$d = 2$	WA	79.1%	91.5%	95.4%	95.7%	93.5%
	AK	60.1%	64.2%	78.7%	84.0%	86.5%
$d = 4$	WA	63.4%	82.9%	85.4%	80.4%	70.9%
	AK	57.2%	58.7%	63.7%	72.2%	70.4%

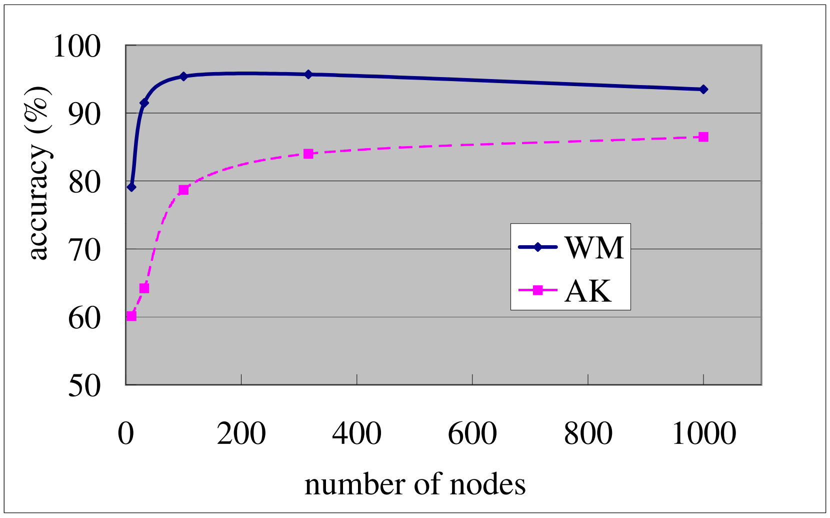

Figure 7: Number of nodes vs Accuracy, where the outdegree is 2.

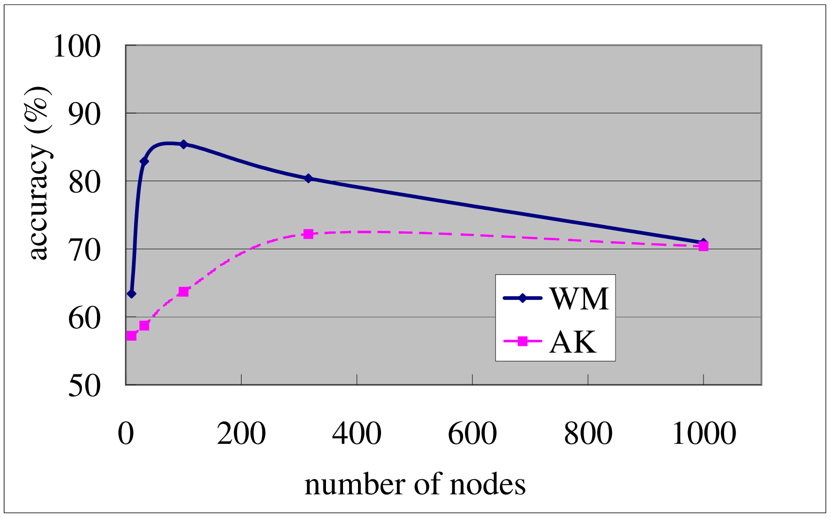

Figure 8: Number of nodes vs Accuracy, where the outdegree is 4.

Acknowledgment

We thank to Osamu Maruyama for fruitful discussion.

References

[1] Akutsu, T., Kuhara, S., Maruyama, O. and Miyano, S., Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions, In *the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 695–702, 1998.

[2] Amaldi, E. and Kann, V., The complexity and approximability of finding maximum feasible subsystem of linear relations, *Theoretical Computer Science*, 147:181–210, 1995.

[3] Berkelaar, M. R. C. M., *lp_solve*, Eindhoven University of Technology, 1996.

[4] Frean, M. R., The upstart algorithm: a method for constructing and training feedforward neural networks, *Neural Comput*, 2:198–209, 1990.

[5] Furukawa, F., Matsumoto, S., Shinohara, A., Shoudai, T. and Miyano, S., HAKKE: A multi-strategy prediction system for sequences, In *Genome Informatics 1996*, pages 98–107, 1996.

[6] Gallant, S. I., Perceptron-based learning algorithms, *IEEE Trans. on Neural Networks*, 1:179–191, 1990.

[7] Marchand, M. and Golea, M., An approximate algorithm to find the largest linearly separable subset of training examples, In *Proc. of 1993 Ann. Meeting of the International Neural Network Society*, pages 556–559, 1993.

[8] Noda, K., Matsumoto, S., Shinohara, A., Shoudai, T. and Miyano, S., Gene finding using HAKKE system, In *Genome Informatics 1997*, pages 318–319, 1997.

[9] Vanderbei, R. J., *LOQO*, Princeton University, 1997.