JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 7, Numbers 3/4, 2000
Mary Ann Liebert, Inc.
Pp. 331–343

Algorithms for Identifying Boolean Networks and Related Biological Networks Based on Matrix Multiplication and Fingerprint Function

TATSUYA AKUTSU,[1] SATORU MIYANO,[1] and SATORU KUHARA[2]

ABSTRACT

Due to the recent progress of the DNA microarray technology, a large number of gene expression profile data are being produced. How to analyze gene expression data is an important topic in computational molecular biology. Several studies have been done using the Boolean network as a model of a genetic network. This paper proposes efficient algorithms for identifying Boolean networks of bounded indegree and related biological networks, where identification of a Boolean network can be formalized as a problem of identifying many Boolean functions simultaneously. For the identification of a Boolean network, an $O(mn^{D+1})$ time naive algorithm and a simple $O(mn^{D})$ time algorithm are known, where n denotes the number of nodes, m denotes the number of examples, and D denotes the maximum indegree. This paper presents an improved $O(m^{\omega-2}n^{D} + mn^{D+\omega-3})$ time Monte-Carlo type randomized algorithm, where ω is the exponent of matrix multiplication (currently, $\omega < 2.376$). The algorithm is obtained by combining fast matrix multiplication with the randomized fingerprint function for string matching. Although the algorithm and its analysis are simple, the result is nontrivial and the technique can be applied to several related problems.

Key words: Boolean network, genetic network, DNA microarray, matrix multiplication, fingerprint function.

1. INTRODUCTION

THE DNA MICROARRAY TECHNOLOGY is one of the most important inventions in recent molecular biology (DeRisi *et al.*, 1997). A lot of projects are starting using the DNA microarray technology. Some of them aim at revealing gene regulation mechanisms from time series of gene expression patterns. Expression profiles of several thousands of genes are now being produced for further analyses.

In order to infer gene regulation mechanism and/or genetic networks from time series of gene expression patterns, many studies have been done. Although clustering has been successfully applied (DeRisi *et al.*, 1997), information produced by clustering is limited and is not enough for reconstructing genetic networks. Therefore, other methods have been studied (Akutsu *et al.*, 1999a; Akutsu *et al.*, 2000; Arkin *et al.*, 1997; Chen *et al.*, 1999; D'haeseleer *et al.*, 1999; Liang *et al.*, 1998).

[1]Human Genome Center, Institute of Medical Science, University of Tokyo, 108-8639, Tokyo, Japan.
[2]Graduate School of Genetic Resources Technology, Kyushu University, 812-8581, Fukuoka, Japan.

Among these studies, several have been done using the *Boolean network* as a model of a genetic network (Akutsu *et al.*, 1999a; Liang *et al.*, 1998). Liang *et al.* (1999) proposed a heuristic algorithm for inference of Boolean networks of bounded indegree. Although they did not analyze the time complexity, it seems that the worst case complexity is $O(mn^{D+1})$, where m is the number of examples, n is the number of nodes, and D is the maximum indegree. We also proposed a simple $O(mn^{D+1})$ time algorithm in order to analyze the sample complexity (Akutsu *et al.*, 1999a). Then we developed an improved $O(mn^{D})$ time algorithm (Akutsu *et al.*, 1999b). On the other hand, we can prove that the identification of a Boolean network is NP-hard if D is not a constant (i.e., D is included in an input). Therefore, it seems very difficult to design an algorithm for which the exponent is much smaller than D. In this paper, we present an improved $O(m^{\omega-2}n^{D} + mn^{D+\omega-3})$ time Monte-Carlo-type randomized algorithm, where ω is the exponent of matrix multiplication and ω is currently less than 2.376 (Coppersmith and Winograd, 1990). The algorithm is obtained by reducing the identification problem to matrix multiplication, using the randomized fingerprint function (Karp and Rabin, 1985). Several related results are shown too.

The technique is also applied to the identification of *functional relations* in a fixed domain and the identification of *qualitative relations*. These problems are also important because they are considered as extensions of Boolean networks. In the identification of functional relations, the binary domain $\{0, 1\}$ is extended to a fixed domain. In the identification of qualitative relations, functions based on differential equations are considered, although only functions with one input variable are considered in this paper. These extensions are important because it was recently recognized that the Boolean network is not sufficient as a model of a genetic network and thus extensions of the Boolean network are becoming important. Moreover, the identification of functional relations may be useful for other biological problems since many biological data are stored in relational databases and extraction of functional relations from the databases is important for analyzing the data.

Although the proposed algorithms are theoretically better than previous ones, they are not efficient in practice because fast matrix multiplication algorithms are not practical. However, these algorithms show the existence of algorithms which are better than the naive algorithms previously developed. We hope that this result may lead to development of faster and practical algorithms.

Since a Boolean network can be considered as a set of Boolean functions, previous algorithms developed for inferring a Boolean function (Kearns and Vazirani, 1994) might be applied to the identification of Boolean networks. In particular, the WINNOW algorithm (Littlestone, 1988) is simple and practical for inferring Boolean functions with a few variables. However, in order to apply the WINNOW algorithm to the identification of Boolean networks of bounded indegree, some postprocessing would be required. It seems that postprocessing will take $O(mn^{D+1})$ time in the worst case, using a simple algorithm, where we are interested in the worst case time complexity in this paper. Of course, the algorithms proposed in this paper can also be used for postprocessing.

Some algorithms were developed for inferring functional relations or *functional dependencies* from relational databases (Mannila and Räihä, 1987; Mannila and Räihä, 1992). Although the developed algorithms are general ones, the worst case time complexities seem to be $O(mn^{D+1})$ if they are modified for functional dependencies with at most D input attributes.

2. PROBLEMS AND RESULTS

2.1. Identification of Boolean functions/networks

In this paper, we consider three types of problems:

CONSISTENCY: Decide whether or not there exists a Boolean network (resp., function) consistent with the given examples, and output one if it exists.

COUNTING: Count the number of Boolean networks (resp., functions) consistent with the given examples.

IDENTIFICATION: Decide whether or not there exists a unique Boolean network (resp., function) consistent with the given examples and output it if it exists.

Although we present algorithms for the *counting problem*, they can be converted for the consistency problem and the identification problem without increasing the order of the time complexity. We formally define the *counting problem* for Boolean networks (resp., functions) as follows.

INPUT: $y_j(k)$ $(j = 1 \ldots l, k = 1 \ldots m)$, $x_i(k)$ $(i = 1 \ldots n, k = 1 \ldots m)$, and integer D, where $x_i(k)$ and $y_j(k)$ take Boolean values (i.e., 0 or 1) respectively,

OUTPUT: for each j, the number of Boolean functions $f_j(x_{i_1}, \ldots, x_{i_D})$'s such that $y_j(k) = f_j(x_{i_1}(k), \ldots, x_{i_D}(k))$ holds for all $k = 1 \ldots m$.

We call a tuple $\langle y_1(k), \ldots, y_l(k), x_1(k), \ldots, x_n(k) \rangle$ for each k an *example.*

This problem is NP-hard for general D (i.e., D is included in INPUT), where the proof is to be given in Section 3.4. Therefore, we are interested in the case where D is a constant. Particularly, we are interested in the case of $D = 2$ since we can reduce higher-dimensional problems to two-dimensional problems by using a simple method described in Section 3.1.3. Moreover, we are interested in the case of $l = 1$ and the case of $l = n$. For the case of $l = 1$, it is the identification problem of a Boolean function. For the case of $l = n$, it is the identification problem of a Boolean network. For a general case of the problem, there is a trivial algorithm: for each y_j, for all types of 2^{2^D} Boolean functions f, for all combinations of D variables $x_{i_1}, \ldots, x_{i_D}$, examine whether or not $y_j(k) = f(x_{i_1}(k), \ldots, x_{i_D}(k))$ holds for $k = 1 \ldots m$. This algorithm takes $O(lmn^D)$ time for a constant D.

Note that, although we should carefully count the number of functions so that the same function is not counted more than once, this can be done without affecting the orders of the time complexities in all algorithms presented in this paper. Here we briefly explain the method, using an example. Consider Boolean functions of the form $x_i \vee x_j$. In this case, the same Boolean function may be counted twice: $x_i \vee x_j$ and $x_j \vee x_i$. However, we can avoid the counting of this Boolean function more than once by considering a total order $x_1 \prec x_2 \prec \ldots \prec x_n$ and counting only $x_i \vee x_j$'s such that $x_i \prec x_j$. This method can be extended for the other types of Boolean functions of $D = 2$. Moreover, this method can be extended for any fixed D in a straightforward way since we assume that D is a fixed constant and thus we only mind fixed types (i.e., 2^{2^D} types) of Boolean functions.

Recently, we developed an improved $O(mn^D + lm)$ time algorithm (Akutsu *et al.*, 1999b), by using a *trie*, which is a well-known data structure in string matching (Aho, 1990). In this paper, we further improve the time complexity and we show the following algorithms:

- an $O(n^D m^{\omega-2} + n^{D+\omega-3}m)$ time deterministic algorithm for the case of $l = 1$,
- an $O(n^D m^{\omega-2} + n^{D+\omega-3}m)$ time Monte-Carlo-type randomized algorithm for the case of $l = n$,

where ω denotes the exponent of matrix multiplication (i.e., matrix multiplication of $n \times n$ matrices can be done in $O(n^\omega)$ time). Note that, for $n \times m$ matrices X and Y, matrix product $X \cdot Y^t$ can be computed in $O(mn^{\omega-1})$ time if $m \geq n$. Otherwise, it can be computed in $O(m^{\omega-2}n^2)$ time, by partitioning each matrix into small square matrices. Note also that, in this paper, Y^t denotes the transposed matrix of Y. Recently, some improvement on matrix multiplication was done for the case of $m \neq n$ (Huang and Pan, 1997). That result might be useful for improving the time complexities of the algorithms for the case of $m \ll n$.

Although Boolean functions are considered in the above, the algorithm for the case of $l = n$ can be extended for finding functional relations (or functional dependencies) in a fixed domain.

2.2. *Identification of qualitative relations*

In order to analyze gene expression data or other time series data, some studies are done on deriving linear differential equations from data (D'haeseleer *et al.*, 1999). However, linear differential equations are not always appropriate. Moreover, in many cases, it is very difficult to decide what kind of differential equations should be used. In such a case, deriving functional relations qualitatively might be useful (Akutsu *et al.*, 2000; Thieffry and Thomas, 1998). It should be noted that *qualitative reasoning* has been studied very well in artificial intelligence (de Kleer and Brown, 1984). Although it is assumed in most previous studies that qualitative rules are given, deriving qualitative rules from data is also important.

In order to develop algorithms for deriving qualitative rules, we must define a qualitative model. But it is not an easy task to develop a satisfactory model, although some efforts have been made (Akutsu *et al.*, 2000; Thieffry and Thomas, 1998). In particular, it seems very difficult to develop a model including qualitative relations with multiple inputs. Therefore, we consider a simple qualitative model which was recently developed (Akutsu *et al.*, 2000). We consider the following simple rules in this paper:

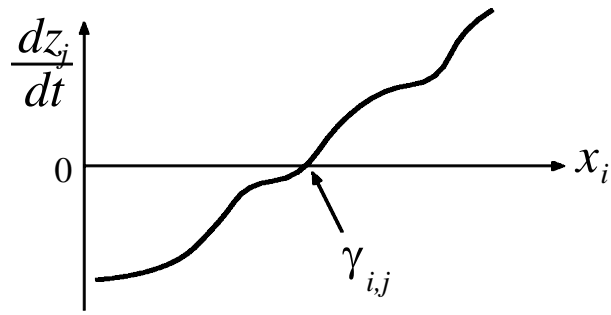

FIG. 1. For a monotonically increasing function $f(x)$, it holds that $dz_j/dt > 0$ iff $x_i > \gamma_{i,j}$, where $dz_j/dt = f(x_i)$.

- x_i activates (resp., inhibits) y_j if the current value (level) of x_i is greater than some threshold value $\gamma_{i,j}$,
- $\frac{dz_j(t)}{dt} > 0$ iff $x_i(t) \geq \gamma_{i,j}$.

These rules are equivalent if $z_j(t)$ expresses the expression level of gene y_j and $\frac{dz_j(t)}{dt} > 0$ means that y_j is activated. These are similar to a threshold function with one input variable.

Note that one-dimensional linear differential equation $\frac{dz(t)}{dt} = ax(t) + b$ satisfies the second rule if $a > 0$ (by letting $\gamma = -b/a$). Of course, we can treat the case of $a < 0$ by replacing $x_i(t) > \gamma_{i,j}$ with $x_i(t) < \gamma_{i,j}$. Moreover, every differential equation $\frac{dz(t)}{dt} = f(x(t))$ also satisfies the second rule if $f(x)$ is a monotonically increasing function (see Figure 1). Based on the above discussion, we define (the 1-d version of) the *counting problem for qualitative relations* as follows.

INPUT: $y_j(k)$ $(j = 1 \ldots l,\ k = 1 \ldots m)$, $x_i(k)$ $(i = 1 \ldots n,\ k = 1 \ldots m)$, where $x_i(k)$ takes a real value and $y_i(k)$ takes either 0 or 1,
OUTPUT: for each y_j, the number of x_i's for which there is a threshold $\gamma_{i,j}$ such that $y_j(k) = 1$ iff $x_i(k) > \gamma_{i,j}$ for all $k = 1 \ldots m$.

The identification problem and the consistency problem can be defined as in Section 2.1. In this definition, "$y_j(k) = 1$" corresponds to "y_j is activated" or "$\frac{dz_j(t)}{dt} > 0$," and "$y_j(k) = 0$" corresponds to "y_j is inhibited" or "$\frac{dz_j(t)}{dt} \leq 0$."

For the case of $l = 1$, there is a simple optimal $O(mn)$ time algorithm: for all i, examine whether or not $(x_i)^+ > (x_i)^-$, where we let $(x_i)^+ = \min\{x_i(k) | y_1(k) = 1\}$ and $(x_i)^- = \max\{x_i(k) | y_1(k) = 0\}$. Note that there exists a threshold $\gamma_{i,1}$ $((x_i)^- \leq \gamma_{i,1} < (x_i)^+)$ iff $(x_i)^+ > (x_i)^-$.

As in Section 2.1, we are particularly interested in the case of $l = n$. Using the algorithm for $l = 1$ repeatedly, we can obtain an $O(mn^2)$ time algorithm. For this case, we show the following improved algorithms:

- an $O(mn \log(mn))$ time Monte-Carlo-type randomized algorithm,
- an $O(mn^{\frac{1+\omega}{2}} + m^{\frac{\omega-1}{2}} n^2)$ time deterministic algorithm.

3. ALGORITHMS FOR BOOLEAN NETWORKS/FUNCTIONS

In this section, we describe algorithms for the counting problem for Boolean functions and Boolean networks. Before describing the algorithms, we note that Boolean functions $f(x, y)$ with two input variables are classified into the following categories:

CONSTANT: 0, 1,
UNARY: x, $\overline{x}$, y, $\overline{y}$,
XOR: $x \oplus y$, $\overline{x \oplus y}$,
AND: $x \wedge y$, $x \wedge \overline{y}$, $\overline{x} \wedge y$, $\overline{x} \wedge \overline{y}$,
OR: $x \vee y$, $x \vee \overline{y}$, $\overline{x} \vee y$, $\overline{x} \vee \overline{y}$.

In this paper, different types of Boolean functions are counted separately. Since counting of CONSTANT and UNARY functions is easier, we consider only AND, OR, XOR functions, where AND functions and OR functions are treated in a similar way based on the fact $\overline{x \vee y} = \overline{x} \wedge \overline{y}$. XOR functions are treated in a different way. It should be noted that, in the counting problem for Boolean functions, counting of XOR functions is easier and faster than counting of AND/OR functions.

3.1. An algorithm for Boolean functions

In this subsection, we consider the case of $l = 1$. First we show an algorithm for counting the number of Boolean functions of the form $x \wedge \overline{y}$. The other types of functions in AND and OR categories can be counted in a similar way.

3.1.1. Counting of $x \wedge \overline{y}$ functions. The algorithm consists of two steps. The first step is similar to the PAC learning algorithm for monotone Boolean functions (Kearns and Vazirani, 1994; Valiant, 1984). It begins with the conjunction of all literals

$$x_1 \wedge x_2 \wedge \ldots \wedge x_n \wedge \overline{x_1} \wedge \overline{x_2} \wedge \ldots \wedge \overline{x_n}$$

and processes examples one by one (from $k = 1$ to $k = m$). If $y_1(k) = 0$, nothing is done. If $y_1(k) = 1$, all x_i's such that $x_i(k) = 0$ and all $\overline{x_i}$'s such that $\overline{x_i(k)} = 0$ (i.e., $x_i(k) = 1$) are deleted from the conjunction respectively. Let $x_{i_1}, \ldots, x_{i_h}, \overline{x_{j_1}}, \ldots, \overline{x_{j_{h'}}}$ be the variables remaining in the conjunction after testing all examples. Let $k_1, k_2, \ldots, k_{m'}$ be the indices such that $y_1(k_i) = 0$.

In the second step, we make two matrices X and Y, where X is the $h \times m'$ integer matrix defined by $X_{t,s} = x_{i_t}(k_s)$ and Y is the $h' \times m'$ integer matrix defined by $Y_{t,s} = \overline{x_{j_t}(k_s)}$. We compute the matrix product $Z = X \cdot Y^t$. Then, we count the number of elements $Z_{i,j}$ such that $Z_{i,j} = 0$. Since $(\forall k)(y_1(k) = x_{i_s}(k) \wedge \overline{x_{j_t}(k)})$ holds iff $Z_{s,t} = 0$, the correct number is output.

Now we analyze the time complexity. Clearly, the first step takes $O(mn)$ time. The second step takes $O(m^{\omega-2}n^2 + mn^{\omega-1})$ time since $m' \leq m$, $h \leq n$ and $h' \leq n$. Therefore, the total time complexity is $O(m^{\omega-2}n^2 + mn^{\omega-1})$.

Example 1. Assume that the following examples are given as an input data set:

y_1	x_1	x_2	x_3	x_4	x_5	x_6
1	1	1	1	0	1	0
0	0	1	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	1	1	0
1	1	1	1	0	0	0
0	0	1	0	0	1	1

After the first step, we have the following table:

y_1	x_1	x_2	x_3	$\overline{x_4}$	$\overline{x_6}$
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	0	1
0	0	1	0	1	0

In the second step, we compute the matrix product:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 1 \\ 0 & 0 \end{pmatrix}$$

Thus, we have the following three Boolean functions consistent with the given examples:

$$y_1(k) = x_1(k) \wedge \overline{x_4(k)},$$
$$y_1(k) = x_3(k) \wedge \overline{x_4(k)},$$
$$y_1(k) = x_3(k) \wedge \overline{x_6(k)}.$$

3.1.2. Counting of $x \oplus y$ functions. This case is easier than the above case and we do not require matrix multiplication. First note that $x \oplus y = z$ iff $x \oplus z = y$. Let $str(x_i)$ be the sequence of Boolean values of $x_i(k)$ (i.e., $str(x_i) = \langle x_i(1), x_i(2), \ldots, x_i(m)\rangle$). Let $str'(x_i)$ be the sequence of Boolean values of $x_i(k) \oplus y_1(k)$ (i.e., $str'(x_i) = \langle x_i(1) \oplus y_1(1), x_i(2) \oplus y_1(2), \ldots, x_i(m) \oplus y_1(m)\rangle$). Then we count the number of pairs (x_i, x_j) such that $str(x_i) = str'(x_j)$. Of course, it would take $O(mn^2)$ time if we used a naive algorithm. But we can reduce the time complexity to $O(mn)$ by constructing a *trie* (Aho, 1990) for $str(x_i)$'s, as in Akutsu *et al.* (1999b). Here, we briefly review the method for the readers. From a set of sequences $\{str(x_1), \ldots, str(x_n)\}$ over $\Sigma = \{0, 1\}$, we construct a trie. Then, for each $str'(x_j)$, we examine whether there exists a sequence $str(x_i)$ such that $str(x_i) = str'(x_j)$ by traversing the trie. Since this traversal can be done in $O(m)$ time for each $str'(x_j)$, the time required for traversal of all strings $str'(x_1), \ldots, str'(x_n)$ is $O(mn)$. Since the trie can be constructed in $O(mn)$ time, $O(mn)$ time is required in total. It is easy to modify the algorithm for counting of $\overline{x \oplus y}$ functions.

Theorem 1. *The counting problem for Boolean functions of two inputs can be solved in $O(m^{\omega-2}n^2 + mn^{\omega-1})$ time.*

3.1.3. Extension to $D > 2$. Using the above-mentioned algorithm, we can develop an $O(m^{\omega-2}n^D + mn^{D+\omega-3})$ time algorithm for any fixed $D \geq 2$. First, we consider a case of $D = 3$. Then we note that

$$f(x, y, z) = (z \wedge f(x, y, 1)) \vee (\overline{z} \wedge f(x, y, 0))$$

holds for any Boolean function $f(x, y, z)$. Thus, we can count the number of Boolean functions $f(x_i, x_j, x_h)$ for fixed x_h by multiplying the number of $f_1(x_i, x_j)$'s such that $y_1(k) = f_1(x_i(k), x_j(k))$ holds for examples with $x_h(k) = 1$ and the number of $f_2(x_i, x_j)$'s such that $y_1(k) = f_2(x_i(k), x_j(k))$ holds for examples with $x_h(k) = 0$. Since this computation can be done for each x_h, the total time complexity becomes

$$O(n) \times O(m^{\omega-2}n^2 + mn^{\omega-1}) = O(m^{\omega-2}n^3 + mn^{\omega}).$$

For general D, we can apply this method recursively. Since an $O(n)$ factor is multiplied per dimension, the total time complexity is $O(m^{\omega-2}n^D + mn^{D+\omega-3})$.

Corollary 1. *The counting problem for Boolean functions of D inputs can be solved in $O(m^{\omega-2}n^D + mn^{D+\omega-3})$ time.*

3.2. An algorithm for Boolean networks

In this subsection, we consider the case of $l = n$ where the technique can be applied to any l.

In addition to matrix multiplication, we use the randomized fingerprint function developed by Karp and Rabin (Karp and Rabin, 1985; Motowani and Raghavan, 1994). Here, we briefly review the function. Let $s = \langle s_1, s_2, \ldots, s_m\rangle$ and $t = \langle t_1, t_2, \ldots, t_m\rangle$ be strings of length m over $\{0, 1\}$, respectively. Let p be a prime number. We define the fingerprint function $F_p(s)$ by

$$F_p(\langle s_1, s_2, \ldots, s_m\rangle) = s_1 \cdot 2^0 + s_2 \cdot 2^1 + \ldots + s_m \cdot 2^{m-1} \bmod p\,.$$

It was shown that, by choosing a prime number less than $\tau = \Theta(cm \log(cm))$ uniformly at random, $Prob(F_p(s) = F_p(t)) \leq \frac{1}{c}$ holds for any $s \neq t$.

For simplicity, we describe the counting algorithm for Boolean functions of the form $x_i \wedge \overline{x_j}$, which can be easily modified for the other functions in AND and OR categories. For each y_i, we compute $F_p(\langle y_i(1), y_i(2), \ldots, y_i(m)\rangle)$. We make two matrices X and Y, where X is the $n \times m$ integer matrix

defined by $X_{i,k} = x_i(k) \cdot 2^{k-1} \bmod p$, and Y is the $n \times m$ integer matrix defined by $Y_{j,k} = \overline{x_j(k)}$. Next, we compute the matrix product $Z = X \cdot Y^t$ under modulo p (i.e., under $GF(p)$). Then, we partition $Z_{i,j}$'s into groups so that each group consists of elements having the same value. For each y_h, we output the number of elements in the group that has the same value as $F_p(y_h)$.

It is easy to see that

$$Z_{i,j} = (x_i(1) \wedge \overline{x_j(1)}) \cdot 2^0 + (x_i(2) \wedge \overline{x_j(2)}) \cdot 2^1 + \ldots + (x_i(m) \wedge \overline{x_j(m)}) \cdot 2^{m-1} \bmod p.$$

Therefore, for any triplet (y_h, x_i, x_j) satisfying $(\forall k)(y_h(k) = x_i(k) \wedge \overline{x_j(k)})$, $F_p(y_h) = Z_{i,j}$ always holds, whereas $F_p(y_h) \neq Z_{i,j}$ holds with high probability for the other triplets (y_h, x_i, x_j). By letting $\tau = \Theta(mn^{3+\alpha} \log(mn^{3+\alpha}))$, the failure probability (i.e., the probability that a false number is output for some y_h) can be made less than $\frac{1}{n^\alpha}$.

In order to treat XOR functions, we compute $Z_{i,j} + Z_{j,i}$ for all pairs (i, j) such that $i < j$ and count the number of y_h's satisfying $F_p(y_h) = Z_{i,j} + Z_{j,i}$. It should be noted that $F_p(y_h) = Z_{i,j} + Z_{j,i}$ always holds if $(\forall k)(y_h(k) = x_i(k) \oplus x_j(k))$. Otherwise, $F_p(y_h) \neq Z_{i,j} + Z_{j,i}$ holds with high probability because $x_i \oplus x_j = (x_i \wedge \overline{x_j}) \vee (\overline{x_i} \wedge x_j)$ and $(x_i \wedge \overline{x_j}) \wedge (\overline{x_i} \wedge x_j) = 0$ hold.

Now we consider the time complexity. Since we assume the standard RAM model in this paper, each arithmetic operation for $O(\log(nm))$ bit integers can be done in constant time. Therefore, we can assume that each operation in $GF(p)$ can be done in constant time. Generation of a random prime number can be done in $O(poly(\log(\tau)))$ time using a Monte-Carlo-type randomized algorithm (Motowani and Raghavan, 1994). Since all known matrix multiplication algorithms are available in any ring (Motowani and Raghavan, 1994), $Z = X \cdot Y^t$ can be computed in $O(m^{\omega-2}n^2 + mn^{\omega-1})$ time. Since the other parts take $O(n^2 \log n + mn)$ time, we have:

Theorem 2. *The counting problem for Boolean networks of $D = 2$ can be solved in $O(m^{\omega-2}n^2 + mn^{\omega-1})$ time with high probability.*

Using the same technique as in Corollary 1, we can extend the algorithm for any fixed D.

Corollary 2. *The counting problem for Boolean networks of fixed D can be solved in $O(m^{\omega-2}n^D + mn^{D+\omega-3})$ time with high probability.*

Note that, in this paper, "high probability" means a probability of at least $1 - \frac{1}{n^c}$ where c is an arbitrarily fixed constant.

Example 2. Assume that the following examples are given as an input data set:

y_1	y_2	x_1	x_2
0	1	1	0
0	0	1	1
1	0	0	1
0	0	0	0
0	1	1	0

Then we compute the following under modulo p:

$$\begin{pmatrix} 2^0 & 2^1 & 0 & 0 & 2^4 \\ 0 & 2^1 & 2^2 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 2^0 + 2^4 \\ 2^2 & 0 \end{pmatrix}.$$

Since $F_p(y_1) = 2^2$ and $F_p(y_2) = 2^0 + 2^4 \pmod p$, we know that $y_1(k) = x_2(k) \wedge \overline{x_1(k)}$ and $y_2(k) = x_1(k) \wedge \overline{x_2(k)}$ (with high probability).

3.3. Algorithms robust for noises

In practice, input data may contain noises. In such a case, Boolean functions $f(x_{i_1}, \ldots, x_{i_D})$ whose errors are less than a threshold K (i.e., $|\{k|y_j(k) \neq f(x_{i_1}(k), \ldots, x_{i_D}(k))\}| < K$) should be output. A trivial algorithm takes $O(mn^{D+1})$ time for the case of $l = n$. Matrix multiplication is also useful in order to count the error. For example, consider Boolean functions of the form $f(x_i, x_j) = x_i \wedge x_j$. First, for each x_i, we compute $n \times m$ integer matrices X_i and $\overline{X_i}$ defined by $(X_i)_{j,k} = x_i(k) \wedge x_j(k)$ and $(\overline{X_i})_{j,k} = \overline{x_i(k) \wedge x_j(k)}$. Next, we compute $Z_i = X_i \cdot \overline{Y}^t + \overline{X_i} \cdot Y^t$ for each x_i, where Y and $\overline{Y}$ are $n \times m$ integer matrices defined by $Y_{j,k} = y_j(k)$ and $\overline{Y}_{j,k} = \overline{y_j(k)}$ respectively. Then, it is easy to see that $(Z_i)_{s,t}$ is equal to the number of errors between $y_t(k)$ and $x_i(k) \wedge x_s(t)$ (i.e., $(Z_i)_{s,t} = |\{k|y_t(k) \neq x_i(k) \wedge x_s(k)\}|$). This method can be generalized to any Boolean function of fixed D as in Section 3.1.3.

Theorem 3. *The number of Boolean functions $f(x_{i_1}, \ldots, x_{i_D})$ whose errors are less than a threshold K can be computed in $O(m^{\omega-2}n^{D+1} + mn^{D+\omega-2})$ time, where the number must be output for each y_j $(j = 1, \ldots, n)$.*

A simple and well-known random sampling technique can also be used, where we cannot count the exact number in this case. Let θm be a threshold. We randomly select $\alpha\theta \ln(mn)$ examples from m examples in the input, where α is a fixed constant, and we assume w.l.o.g. that $\alpha\theta \ln(mn)$ is an integer. Let S be the set of indices k of randomly selected examples. Then we output, for each y_j, all Boolean functions $f(x_{i_1}, \ldots, x_{i_D})$ satisfying $|\{k \in S|y_j(k) \neq f(x_{i_1}, \ldots, x_{i_D})\}| \leq (1 + \epsilon)\alpha\theta \ln(mn)$. Clearly, this algorithm works in $O(n^{D+1} \log(mn))$ time.

Now we analyze the failure probability. First we analyze the probability that the error of a Boolean function $y_j = f(x_{i_1}, \ldots, x_{i_D})$ against input data is at most θm but the error against S is more than $(1+\epsilon)$ $\alpha\theta \ln(mn)$. From the Chernoff bound (Motowani and Raghavan, 1994), this probability is bounded by

$$Prob(\#ERROR > (1 + \epsilon)\alpha\theta \ln(mn)) < \exp\left(\frac{-\epsilon^2\alpha\theta \ln(mn)}{4}\right) = \frac{1}{(mn)^{\frac{\epsilon^2\alpha\theta}{4}}},$$

where we assume that $0 < \epsilon < 2e - 1$.

Next we analyze the probability that the error of a Boolean function $y_j = f(x_{i_1}, \ldots, x_{i_D})$ against input data is at least $(1 + 2\epsilon)\theta m$ but the error against S is at most $(1 + \epsilon)\alpha\theta \ln(mn)$. Using the Chernoff bound again, this probability is bounded by

$$Prob(\#ERROR \leq (1 + \epsilon)\alpha\theta \ln(mn)) < \exp\left(-\frac{\left(\frac{\epsilon}{1+2\epsilon}\right)^2 (1 + 2\epsilon)\alpha\theta \ln(mn)}{2}\right) = \frac{1}{(mn)^{\frac{\epsilon^2\alpha\theta}{2(1+2\epsilon)}}}$$

for $\epsilon > 0$. Therefore, by choosing sufficiently large α, we have:

Theorem 4. *There exists an $O(n^{D+1} \log(mn))$ time randomized algorithm that outputs, for each y_j, all Boolean functions whose errors are at most θm with high probability and does not output, for each y_j, any Boolean function whose error is at least $(1 + 2\epsilon)\theta m$ with high probability, where $0 < \theta < 1$ and $0 < \epsilon < 2e - 1$ are fixed constants.*

It should be noted that neither algorithm is yet practical because each algorithm takes more than $O(n^3)$ time even for $D = 2$. Recall that even *Saccharomyces cerevisiae* (budding yeast) has approximately 6000 genes (DeRisi *et al.*, 1997) (i.e., $n \approx 6000$).

3.4. An approximation algorithm

We have been considering the counting problem for a fixed D since the problem is NP-hard if D is not a constant. This NP-hardness result can be proven even for the case of $l = 1$.

Theorem 5. *The counting problem for Boolean functions is NP-hard for general D.*

Proof. We use a polynomial time reduction from SET COVER, where similar reductions are used by Akutsu and Bao (1996), Makino (1997), and Mannila and Räihä (1992). Let $(\mathcal{S} = \{S_1, S_2, \ldots, S_n\}, D)$ be an instance of SET COVER over $U = \{u_1, \ldots, u_m\}$. Recall that SET COVER is to decide whether or not there exist D-sets $S_{i_1}, \ldots, S_{i_D}$ such that $S_{i_1} \cup \ldots \cup S_{i_D} = U$. From $(\mathcal{S}, D)$, we construct $m + 1$ examples as follows:

$$\begin{aligned} y_1(k) &= 1, \quad \text{for } k = 1, \ldots, m, \\ y_1(m+1) &= 0, \\ x_i(k) &= 1, \quad \text{if } k \le m \text{ and } u_k \in S_i, \\ x_i(k) &= 0, \quad \text{otherwise.} \end{aligned}$$

Then there exists a Boolean function $f(x_{i_1}, \ldots, x_{i_D})$ such that $(1 \le \forall k \le m+1)(y_1(k) = f(x_{i_1}(k), \ldots, x_{i_D}(k))$ if and only if $S_{i_1} \cup \ldots \cup S_{i_D} = U$. ■

SET COVER can also be used for developing an approximation algorithm. As in Akutsu and Bao (1996), we reduce the problem to the SET COVER problem, where SET COVER is defined as a minimization problem (i.e., finding $S_{i_1}, \ldots, S_{i_D}$ with minimum D) in this case. From the given examples, we construct U and S_i's by

$$\begin{aligned} U &= \{(k, k') | k < k' \text{ and } y_j(k) \ne y_j(k')\}, \\ S_i &= \{(k, k') | x_i(k) \ne x_i(k')\} \cap U. \end{aligned}$$

Then we apply an approximation algorithm for SET COVER (Johnson, 1974) to S_i's and U. Since SET COVER can be approximated within a factor of $\ln |U| + 1$, we have:

Theorem 6. *Assume that $f_j(x_{i_1}(k), \ldots, x_{i_D}(k)) = y_j(k)$ holds for all k. Then a set of variables $\{x_{i'_1}, \ldots, x_{i'_h}\}$ for which $h \le (2 \ln m + 1)D$ holds and there exists a Boolean function f'_j satisfying $f'_j(x_{i'_1}(k), \ldots, x_{i'_h}(k)) = y_j(k)$ for all k can be found in polynomial time.*

Although a set of variables can be found in polynomial time, it seems difficult to determine f'_j in polynomial time because there exist 2^{2^d} Boolean functions with d input variables. It should be noted that description of a function needs $\Omega(2^d)$ space unless types of Boolean functions are restricted.

3.5. An algorithm for finding functional relations

Although the domain of values is restricted to $\{0, 1\}$ in Boolean networks, the algorithm in Section 3.2 can be extended for other fixed-size domains. Since Boolean values may not be adequate for representing gene expression levels, this extension is important. As in Section 3.2, we explain the algorithm for the case of $D = 2$. Extension to other D's can be done as in Section 3.1.3.

Let Σ be the domain (i.e., $x_i(k), y_h(k) \in \Sigma$), where we let $b = |\Sigma|$. In this case, we use the fingerprint function on base b:

$$F_{p,b}(\langle s_1, s_2, \ldots, s_m \rangle) = s_1 \cdot b^0 + s_2 \cdot b^1 + \ldots + s_m \cdot b^{m-1} \bmod p\,,$$

where it is known that a property similar to that of F_p holds for this function (Motowani and Raghavan, 1994). For each function f in $\Sigma \times \Sigma \to \Sigma$, we examine whether or not there exists a triplet (y_h, x_i, x_j) such that $y_h(k) = f(x_i(k), x_j(k))$ holds for all k. For each $\alpha \in \Sigma$, let X^α be the $n \times m$ matrix defined by

$$X^\alpha_{i,k} = \begin{cases} 1, & \text{if } x_i(k) = \alpha, \\ 0, & \text{otherwise.} \end{cases}$$

For each $\alpha \in \Sigma$, let Y^α be the $n \times m$ matrix defined by $Y^\alpha_{j,k} = f(\alpha, x_j(k))$, where we encode each element in Σ by using an element in $\{0, 1, \ldots, b-1\}$. Let $Z = \sum_\alpha X^\alpha \cdot (Y^\alpha)^t$. Then, $F_{p,b}(\langle y_h(1), \ldots, y_h(m) \rangle) = Z_{i,j}$ holds if $y_h(k) = f(x_i(k), x_j(k))$ holds for all k, and $F_{p,b}(\langle y_h(1), \ldots, y_h(m) \rangle) \ne Z_{i,j}$ holds with high probability if $y_h(k) \ne f(x_i(k), x_j(k))$ holds for some k.

Theorem 7. *For a fixed domain, the counting problem for functional relations of fixed D can be solved in $O(m^{\omega-2}n^D + mn^{D+\omega-3})$ time with high probability.*

4. ALGORITHMS FOR FINDING QUALITATIVE RELATIONS

4.1. Simple deterministic algorithms

As mentioned in Section 2.2, there is a simple $O(mn^2)$ time deterministic algorithm for the counting problem for qualitative relations.

Proposition 1. *The counting problem for qualitative relations can be solved in $O(mn^2)$ time.*

Here, we present an $O(m^2n)$ time deterministic algorithm. Let $str(y_i)$ be the sequence of $y(i)$'s (i.e., $str(y_i) = \langle y(1), y(2), \ldots, y(m)\rangle$). Then, the following is the description of the algorithm.

1. Make a trie for the set of sequences $\{str(y_1), str(y_2), \ldots, str(y_n)\}$.
2. For all x_i, sort $x_i(k)$'s in the increasing order and let $\hat{x}_i(1) < \hat{x}_i(2) < \ldots < \hat{x}_i(m)$ be the sorted sequence, where we assume w.l.o.g. $x_i(k) \neq x_i(k')$ for $k \neq k'$.
3. For all x_i and for all $h \in \{0, 1, \ldots, m\}$, construct the sequence

$$str(x_i^h) = \langle sign(x_i(1), h), sign(x_i(2), h), \ldots, sign(x_i(m), h)\rangle ,$$

where

$$sign(x_i(k), h) = \begin{cases} 1, & \text{if } x_i(k) > \hat{x}_i(h), \\ 0, & \text{otherwise,} \end{cases}$$

and we let $\hat{x}_i(0) = -\infty$.
4. For all $str(y_j)$, count the number of $str(x_i^h)$'s such that $str(y_j) = str(x_i^h)$.

Example 3. Let $x_i(1 \ldots 5) = (5, 2, 3, 1, 4)$ and let $y_j(1 \ldots 5) = (1, 0, 0, 0, 1)$. Then, $str(x_i^h(k))$'s are as follows:

$$str(x_i^0) = \langle 1, 1, 1, 1, 1\rangle,$$

$$str(x_i^1) = \langle 1, 1, 1, 0, 1\rangle,$$

$$str(x_i^2) = \langle 1, 0, 1, 0, 1\rangle,$$

$$str(x_i^3) = \langle 1, 0, 0, 0, 1\rangle,$$

$$str(x_i^4) = \langle 1, 0, 0, 0, 0\rangle,$$

$$str(x_i^5) = \langle 0, 0, 0, 0, 0\rangle.$$

In this case, the threshold $\gamma_{i,j}$ is between $\hat{x}_i(3)$ and $\hat{x}_i(4)$ because $str(x_i^3) = str(y_j)$.

The correctness of the algorithm is obvious. Since Step 3 and Step 4 take $O(m^2n)$ time, the total time complexity is $O(m^2n)$.

Proposition 2. *The counting problem for qualitative relations can be solved in $O(m^2n)$ time.*

4.2. A randomized algorithm

In order to develop a nearly optimal randomized algorithm, we modify the $O(m^2n)$ time algorithm, using the randomized fingerprint function F_p. The modification is very simple and is based on the following observation: when $F_p(str(x_i^h))$ is known, $F_p(str(x_i^{h+1}))$ can be computed in constant time by

$$F_p(str(x_i^{h+1})) = F_p(str(x_i^h)) - x_i(k) \cdot 2^{k-1} \bmod p\,,$$

where $x_i(k)$ is the element at which $str(x_i^{h+1})$ differs from $str(x_i^h)$. The following is the description of the randomized algorithm.

1. For all y_j, compute $F_p(str(y_j))$.
2. For all x_i and for all $h \in \{0, 1, \ldots, m\}$, compute $F_p(str(x_i^h))$.
3. For all $str(y_j)$, count the number of $str(x_i^h)$'s such that $F_p(y_j) = F_p(str(x_i^h))$.

Here we consider the time complexity. Clearly, Step 1 takes $O(mn)$ time. From the observation mentioned above, Step 2 takes $O(mn)$ time too. Step 3 takes $O(mn\log(mn))$ time by using an appropriate search tree for $F_p(str(x_i^h))$'s. Therefore, the total time complexity is $O(mn\log(mn))$.

As in the case of Boolean networks, the failure probability can be made less than $\frac{1}{n^\alpha}$ for any fixed constant $\alpha \geq 1$ under the standard RAM model, without increasing the order of the time complexity. Therefore, we have:

Theorem 8. *The counting problem for qualitative relations can be solved in $O(mn\log(mn))$ time with high probability.*

4.3. A deterministic algorithm

Although we have not yet succeeded in developing a nearly optimal deterministic algorithm for the counting problem for qualitative relations, we can develop an $o(mn^2)$ time deterministic algorithm.

Let $*$ denote the "don't care" character. That is, $*$ matches any single character. Then the following lemma is obtained by combining matrix multiplication with the idea used in pattern matching with don't care characters (Fisher and Paterson, 1974).

Lemma 1. *Given strings $s^1, \ldots, s^n$ over an alphabet $\{0, 1, *\}$ and strings $t^1, \ldots, t^n$ over an alphabet $\{0, 1\}$ each of which has length m, all pairs (i, j) such that s^i matches t^j can be enumerated in $O(mn^{\omega-1} + m^{\omega-2}n^2)$ time.*

Proof. Let $s^i(k)$ (resp., $t^j(k)$) be the k-th letter of s^i (resp., t^j). Then, s^i matches t^j if and only if there does not exist k such that $s^i(k) = 0 \wedge t^j(k) = 1$ or $s^i(k) = 1 \wedge t^j(k) = 0$. This condition can be checked for all s^i, t^j by using matrix multiplication as in Section 3.1.1. ■

The algorithm consists of two parts: rough matchings are first computed and then exact matchings are computed, where a similar technique was already used for pattern matching of 2-D figures (Amir and Farach, 1991). In order to find rough matchings, for each x_i, we make $M + 1$ strings containing don't care characters (see Figure 2), where M is to be determined later and we assume w.l.o.g. that M divides m. We define $\widetilde{str}(x_i^h)$ by

$$\widetilde{str}(x_i^h) = \langle \widetilde{sign}(x_i^h(1)), \widetilde{sign}(x_i^h(2)), \ldots, \widetilde{sign}(x_i^h(m)) \rangle$$

for $h = 0, \frac{m}{M}, \frac{2m}{M}, \ldots, \frac{m(M-1)}{M}$, where

$$\widetilde{sign}(x_i^h(k)) = \begin{cases} 0, & \text{if } x_i(k) \leq \hat{x}_i(h), \\ *, & \text{if } \hat{x}_i(h) < x_i(k) \leq \hat{x}_i(h + \frac{m}{M}), \\ 1, & \text{if } x_i(k) > \hat{x}_i(h + \frac{m}{M}). \end{cases}$$

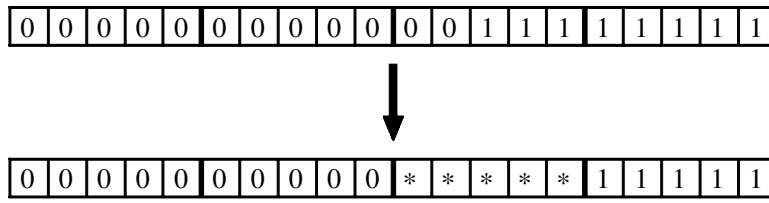

FIG. 2. Construction of $\widetilde{str}(x_i^h)$ for $h = 10$, where $m = 20$, $M = 4$, and $x_i(1) < x_i(2) < \ldots < x_i(m)$ in this example.

Then the following is the description of the algorithm.

1. For all $\widetilde{str}(x_i^h)$ and for all $str(y_j)$, examine whether or not $\widetilde{str}(x_i^h)$ matches $str(y_j)$.
2. For all pairs found in Step 1, examine whether any one of $str(x_i^h)$, $str(x_i^{h+1})$, ..., $str(x_i^{h+\frac{m}{M}-1})$ matches $str(y_j)$.

Here we analyze the time complexity. First we assume $m \geq n$. Clearly, Step 1 can be done in $O(Mmn^{\omega-1})$ time, using Lemma 1. Note that for each pair (x_i, y_j), at most one $\widetilde{str}(x_i^h)$ matches $str(y_j)$. Moreover, in Step 2, we need to consider only a part of $\widetilde{str}(x_i^h)$ that consists of don't care characters. That is, we examine only exact matches of two strings with length m/M for at most n^2 pairs. From Proposition 1, it can be done in $O((m/M)n^2)$ time. Therefore, the total time complexity is $O(Mmn^{\omega-1} + (m/M)n^2))$. By letting $M = n^{\frac{3-\omega}{2}}$, the time complexity becomes $O(mn^{\frac{1+\omega}{2}})$.

For the case of $m < n$, the total time complexity is $O(Mm^{\omega-2}n^2 + (m/M)n^2))$ because Step 1 takes $O(Mm^{\omega-2}n^2)$ time. By letting $M = m^{\frac{3-\omega}{2}}$, the time complexity becomes $O(m^{\frac{\omega-1}{2}}n^2)$.

Theorem 9. *The counting problem for qualitative relations can be solved in* $O(mn^{\frac{1+\omega}{2}} + m^{\frac{\omega-1}{2}}n^2)$ *time.*

5. CONCLUDING REMARKS

In this paper, we presented improved algorithms for identification of Boolean networks and related biological networks. Although most of the proposed algorithms are not efficient in practice, the results show the existence of algorithms which are better than the naive algorithms.

If an ultimate matrix multiplication algorithm ($\omega = 2$?) were developed, the time complexity of the identification algorithm for a Boolean network of $D = 2$ would be $O(n^2 \log n + mn)$, which is nearly optimal in the case of $m \geq n$. However, it is still far from optimal when $m \ll n$. Therefore, development of faster algorithms, in particular, development of an algorithm for which the exponent of n is less than 2 (for $D = 2$) is an open problem. In the identification of functional relations, we assumed a fixed domain. Development of an $o(mn^D)$ time algorithm for any domain is also an open problem.

For the identification of qualitative relations, we considered only functions with one input variable. However, functions with multiple input variables should be treated. Although we have been trying to make a qualitative model including functions with multiple inputs (Akutsu *et al.*, 2000), we have not yet succeeded in developing a satisfactory model. Therefore, development of such a model is important future work.

ACKNOWLEDGMENTS

This work was partially supported by a grant-in-aid for "Genome Science" from the Ministry of Education, Science, Sports and Culture in Japan.

REFERENCES

Aho, A.V. 1990. Algorithms for finding patterns in strings. In Van Leeuwen, J. ed., *Handbook of Theoretical Computer Science*, Vol. A, Elsevier Science, Amsterdam.

Akutsu, T., and Bao, F. 1996. Approximating minimum keys and optimal substructure screens, 290–299. In *Proc. 2nd Int. Conf. Computing and Combinatorics.* Lecture Notes in Computer Science, 1090, Springer-Verlag.

Akutsu, T., Miyano, S., and Kuhara, S. 1999a. Identification of genetic networks from a small number of gene expression patterns under the Boolean network model, 17–28. In *Proc. Pacific Symposium on Biocomputing '99*, World Scientific, Singapore.

Akutsu, T., Miyano, S., and Kuhara, S. 1999b. Fast identification of Boolean networks, 25–32. In Technical Report 99-AL-66, Information Processing Society of Japan.

Akutsu, T., Miyano, S., and Kuhara, S. 2000. Algorithms for inferring qualitative models of biological networks, 293–304. In *Proc. Pacific Symposium on Biocomputing 2000*, World Scientific, Singapore.

Amir, A., and Farach, M. 1991. Efficient two-dimensional approximate matching of nonrectangular figures, 212–223. In *Proc. ACM-SIAM Symp. on Discrete Algorithms*, ACM Press.

Arkin, A., Shen, P., and Ross, J. 1997. A test case of correlation metric construction of a reaction pathway from measurements. *Science* 277, 275–1279.

Chen, T., Filkov, V., and Skiena, S.S. 1999. Identifying gene regulatory networks from experimental data, 94–103. In *Proc. 3rd Int. Conf. on Computational Molecular Biology*, ACM Press.

Coppersmith, D., and Winograd, S. 1990. Matrix multiplication via arithmetic progression. *J. Symbolic Computation* 9, 251–280.

de Kleer, J., and Brown, J.S. 1984. Qualitative physics based on confluences. *Artificial Intelligence* 24, 7–83.

DeRisi, J.L., Lyer, V.R., and Brown, P.O. 1997. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science* 278, 680–686.

D'haeseleer, P., Wen, X., Fuhrman, S., and Somogyi, R. 1999. Linear modeling of mRNA expression levels during CNS development and injury, 41–52. In *Proc. Pacific Symposium on Biocomputing'99*, World Scientific, Singapore.

Fisher, M.J., and Paterson, M.S. 1974. String matching and other products, 113–125. In *Proc. SIAM-AMS Conference on Complexity of Computation*, AMS.

Huang, X., and Pan, V. 1997. Fast rectangular matrix multiplication, 11–23. In *Proc. ACM Symp. Parallel Algebraic and Symbolic Computation*, ACM Press.

Johnson, D.S. 1974. Approximation algorithms for combinatorial problems. *J. Computer and System Sciences* 9, 256–278.

Karp, R.M., and Rabin, M.O. 1985. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development* 31, 249–260.

Kearns, M.J., and Vazirani, U.V. 1994. *An Introduction to Computational Learning Theory.* The MIT Press, Cambridge, MA.

Liang, S., Fuhrman, S., and Somogyi, R. 1998. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures, 18–29. In *Proc. Pacific Symposium on Biocomputing '98*, World Scientific, Singapore.

Littlestone, N. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* 2, 285–318.

Makino, K. 1997. *Studies on Positive and Horn Boolean Functions with Applications to Data Analysis.* Ph.D Thesis, Kyoto University.

Mannila, H., and Räihä, K. 1987. Dependency inference, 155–158. In *Proc. 13th VLDB Conference.*

Mannila, H., and Räihä, K. 1992. On the complexity of inferring functional dependencies. *Discrete Applied Mathematics* 40, 237–243.

Motowani, R., and Raghavan, P. 1994. *Randomized Algorithms.* Cambridge University Press, New York.

Thieffry, D., and Thomas, R. 1998. Qualitative analysis of gene networks, 77–88. In *Proc. Pacific Symposium on Biocomputing '98*, World Scientific, Singapore.

Valiant, L.G. 1984. A theory of the learnable. *Communications of the ACM* 27, 1134–1142.

Address correspondence to:
Tatsuya Akutsu
Human Genome Center
Institute of Medical Science
University of Tokyo
108-8639, Tokyo, Japan

E-mail: tatutsu@ims.u-tokyo.ac.jp

This article has been cited by:

1. Daizhan Cheng, Yin Zhao. 2011. Identification of Boolean control networks. *Automatica* **47**:4, 702-710. [CrossRef]
2. R. Porreca, E. Cinquemani, J. Lygeros, G. Ferrari-Trecate. 2010. Identification of genetic network dynamics with unate structure. *Bioinformatics* **26**:9, 1239-1245. [CrossRef]
3. André Fujita, João Ricardo Sato, Marcos Angelo Almeida Demasi, Satoru Miyano, Mari Cleide Sogayar, Carlos Eduardo FerreiraAn Introduction to Time-Varying Connectivity Estimation for Gene Regulatory Networks 205-230. [CrossRef]
4. Jean-Philippe VertReconstruction of Biological Networks by Supervised Machine Learning Approaches 163-188. [CrossRef]
5. Aaron L. Vollrath, Christopher A. BradfieldAn Introduction to Toxicogenomics . [CrossRef]
6. F. Mordelet, J.-P. Vert. 2008. SIRENE: supervised inference of regulatory networks. *Bioinformatics* **24**:16, i76-i82. [CrossRef]
7. Xiaogang Ruan, Jinlian Wang, Hui Li, Rhoda E. Perozzi, Edmund F. Perozzi. 2008. The use of logic relationships to model colon cancer gene expression networks with mRNA microarray data#. *Journal of Biomedical Informatics* **41**:4, 530-543. [CrossRef]
8. Wenbin Liu, Harri Lähdesmäki, Edward R. Dougherty, Ilya Shmulevich. 2008. Inference of Boolean Networks Using Sensitivity Regularization. *EURASIP Journal on Bioinformatics and Systems Biology* **2008**, 1-12. [CrossRef]
9. Shinn-Ying Ho, Chih-Hung Hsieh, Fu-Chieh Yu, Hui-Ling Huang. 2007. An Intelligent Two-Stage Evolutionary Algorithm for Dynamic Pathway Identification From Gene Expression Profiles. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **4**:4, 648-704. [CrossRef]
10. A. Fujita, J.R. Sato, H.M. Garay-Malpartida, P.A. Morettin, M.C. Sogayar, C.E. Ferreira. 2007. Time-varying modeling of gene expression regulatory networks using the wavelet dynamic vector autoregressive method. *Bioinformatics* **23**:13, 1623-1630. [CrossRef]
11. Bor-Sen Chen, Cheng-Wei Li. 2007. Analysing microarray data in drug discovery using systems biology. *Expert Opinion on Drug Discovery* **2**:5, 755-768. [CrossRef]
12. John Rachlin, Dikla Dotan Cohen, Charles Cantor, Simon Kasif. 2006. Biological context networks: a mosaic view of the interactome. *Molecular Systems Biology* **2**. . [CrossRef]
13. Dougu Nam, Seunghyun Seo, Sangsoo Kim. 2006. An efficient top-down search algorithm for learning Boolean networks of gene expression. *Machine Learning* **65**:1, 229-245. [CrossRef]
14. Isaac S. Kohane, Atul ButteBioinformatics in Functional Genomics . [CrossRef]
15. Kathleen Marchal, Kristof Engelen, Frank De Smet, Bart De MoorComputational Biology and Toxicogenomics 37-92. [CrossRef]
16. Hiroyoshi Toyoshiba, Takeharu Yamanaka, Hideko Sone, Frederick M. Parham, Nigel J. Walker, Jeanelle Martinez, Christopher J. Portier. 2004. Gene Interaction Network Suggests Dioxin Induces a Significant Linkage between Aryl Hydrocarbon Receptor and Retinoic Acid Receptor Beta. *Environmental Health Perspectives* **112**:12, 1217-1224. [CrossRef]
17. John Goutsias, Seungchan Kim. 2004. A Nonlinear Discrete Dynamical Model for Transcriptional Regulation: Construction and Properties. *Biophysical Journal* **86**:4, 1922-1945. [CrossRef]
18. Yuk Fai Leung, Duccio Cavalieri. 2003. Fundamentals of cDNA microarray data analysis. *Trends in Genetics* **19**:11, 649-659. [CrossRef]
19. HIRO TAKAHASHI, SHUTA TOMIDA, TAKESHI KOBAYASHI, HIROYUKI HONDA. 2003. Inference of Common Genetic Network Using Fuzzy Adaptive Resonance Theory

Associated Matrix Method. *Journal of Bioscience and Bioengineering* **96**:2, 154-160. [CrossRef]

20. Alberto de la Fuente, Paul Brazhnik, Pedro Mendes. 2002. Linking the genes: inferring quantitative gene networks from microarray data. *Trends in Genetics* **18**:8, 395-398. [CrossRef]
21. Arthur L Castle, Michael P Carver, Donna L Mendrick. 2002. Toxicogenomics: a new revolution in drug safety. *Drug Discovery Today* **7**:13, 728-736. [CrossRef]
22. Marc Ladanyi, Wing C. Chan, Timothy J. Triche, William L. Gerald. 2001. Expression Profiling of Human Tumors: The End of Surgical Pathology?. *The Journal of Molecular Diagnostics* **3**:3, 92-97. [CrossRef]
23. P Meltzer. 2001. Spotting the target: microarrays for disease gene discovery. *Current Opinion in Genetics & Development* **11**:3, 258-263. [CrossRef]