アルゴリズム 66－4
（1999. 1. 27）

ブーリアンネットワークの高速同定アルゴリズム

阿久津 達也* 宮野 悟* 久原 哲#

*東京大学医科学研究所ヒトゲノム解析センター

#九州大学大学院生物資源環境科学研究科

ブーリアンネットワークは遺伝子の制御ネットワークをモデル化したもので、複雑系分野などで多くの研究がなされてきた。一方、最近、DNAマイクロアレイなどの実験技術の進展により、遺伝子の発現パターンの変化を観察することにより遺伝子の制御ネットワークを推定しようという試みが盛んに行なわれるなってきており、そのための種々の推定アルゴリズムも提案されつつある。Liang, Fuhrman, Somogyiはブーリアンネットワークによるモデル化のもとでのネットワーク同定アルゴリズムを開発したが、その効率化は課題として残されていた。本稿ではより効率的な同定アルゴリズムを示すとともに、制約されたネットワークにおいて平均的に更に高速に動作するアルゴリズムも示す。

Fast Identification of Boolean Networks

Tatsuya Akutsu* Satoru Miyano* Satoru Kuhara#

*Human Genome Center, Institute of Medical Science, University of Tokyo
4-6-1 Shirkanedai, Minato-ku, Tokyo 108-8639, Japan
{takutsu,miyano}@ims.u-tokyo.ac.jp

Graduate School of Genetic Resources Technology, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan
kuhara@grt.kyushu-u.ac.jp

In molecular biology, a lot of projects are starting using the DNA microarray technology. Some of them aim at revealing gene regulation mechanism from time series of gene expression patterns. For such purpose, several algorithms for inferring genetic network architectures have been proposed. Liang, Fuhrman and Somogyi have proposed a heuristic algorithm for inferring genetic networks from state transition tables which correspond to time series of gene expression patterns, using the Boolean network model. However, the time complexities of the algorithm were rather high. This paper proposes faster algorithms.

1 Introduction

In molecular biology, a lot of projects are starting using the DNA microarray technology. Some of them aim at revealing gene regulation mechanism from time series of gene expression patterns [5, 13, 14]. 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, some methods have been developed. Arkin, Shen and Ross [4] proposed a statistical method using correlation matrices to infer chemical reaction networks from time series of measured concentration of species. However, it seems difficult to apply their method to the inference of large scale genetic networks. DeRisi, Iyer and Brown [5] inferred a metabolic pathway from gene expression patterns of *Saccharomyces cerevisiae* obtained by using DNA microarrays. Yuh, Bolouri and Davidson [14] constructed a network model similar to the Boolean network model from time series of expression patterns relating to a sea urchin gene. However, these inference methods are not systematic or automatic.

On the other hand, some studies have been done on the inference of genetic networks from state transition data using the *Boolean network* model [10] and its variants [2, 11]. Recently, Liang, Fuhrman and Somogyi [7] proposed an algorithm named REVEAL (REVerse Engineering ALgorithm) for inference of Boolean networks (corresponding to genetic networks) from state transition tables (corresponding to time series of gene expression patterns). The results of computational experiments on REVEAL suggested that only a small number of state transition pairs (100 pairs from 10^{15}) were sufficient for inferring Boolean networks with 50 nodes (genes) whose *indegree* (the number of input nodes to a node) was bounded by 3.

Independently, we have investigated strategies for identifying genetic networks from gene expression patterns derived by gene disruptions and gene overexpressions using a Boolean network-like model [2]. In that study, we have not assumed that time series of expression patterns are observable [2] and thus the derived bounds on experimental complexity are too high to be practical if it would be applied directly. However, the recent progress of DNA microarray is making it possible to observe time series of gene expression patterns. Thus, we have studied the number of gene expression patterns required to identify the genetic network [3] using the standard Boolean network model [10]. We proposed a simple algorithm for identifying the original Boolean network from the state transition pairs and analyzed the number of expression patterns required to identify the network uniquely. We proved that $\Theta(\log n)$ transition pairs are necessary and sufficient in order to identify the original Boolean network of n nodes uniquely with a high probability if the maximum indegree is bounded by a constant and transition pairs are given uniformly at random from 2^n possible pairs, where $\log x$ means $\log_2 x$ throughout this paper.

Although the algorithm proposed by us was very simple, it is not so efficient. It takes $O(n^{K+1}m)$ time for identifying a Boolean network of maximum indegree K, where m denotes the number of state transition pairs. Note that it takes $O(n^3m)$ time even for a case of $K = 2$. Thus, we left improvement of the efficiency as future work [3]. Liang, Fuhrman and Somogyi also considered the problem of the efficiency [7]. In oder to reduce the search space, REVEAL used information theoretic principles. However, they did not make theoretical analysis and the time complexity of REVEAL is not clear. It seems that the worst case complexity is still $O(n^{K+1}m)$. They also pointed out that improvement of the efficiency was important.

In this paper, we propose an improved algorithm (denoted by IDENT1) whose time complexity is $O(n^Km)$ for fixed K. This algorithm is simple and thus practical. We also propose another algorithm (denoted by IDENT2) which runs in nearly $O(n^Km)$ time in the average case, where we assume that state transition pairs are given at random and $m = \Omega(\log n)$. Moreover, IDENT2 runs in $O(n^2m)$ time in the average case for fixed K if Boolean functions are restricted to AND/OR of literals.

Although real genetic networks are different from the Boolean network, extensions of the Boolean network and the identification algorithms are discussed in Refs. [3, 7]. Therefore, the ideas proposed in this paper might be useful for developing identification algorithms for more realistic models of genetic networks.

2 Boolean Network and Simple Identification Algorithm

2.1 Boolean Network

A *Boolean network* $G(V, F)$ consists of a set $V = \{v_1, \ldots, v_n\}$ of nodes representing genes and a list $F = (f_1, \ldots, f_n)$ of *Boolean functions*, where a Boolean function $f_i(v_{i_1}, \ldots, v_{i_k})$ with inputs from specified nodes $v_{i_1}, \ldots, v_{i_k}$ is assigned to each node v_i. For a subset $U \subseteq V$, an *expression pattern* ψ of U is a function from U to $\{0, 1\}$. An expression pattern of V is also called a *state* of a Boolean network. That is, ψ represents the states of nodes (genes), where each node is assumed

to take either 0 (not-express) or 1 (express) as its state value. If it does not cause confusion, we omit ψ. In a Boolean network, the expression pattern ψ_{t+1} at time $t+1$ is determined by Boolean functions F from the expression pattern ψ_t at time t (i.e., $\psi_{t+1}(v_i) = f_i(\psi_t(v_{i_1}), \ldots, \psi_t(v_{i_k}))$).

It is convenient to consider a *wiring diagram* $G'(V', F')$ of a Boolean network $G(V, F)$ (see Fig. 1) [7, 10]. For each node v_i in V, let $v_{i_1}, \ldots, v_{i_k}$ be input nodes to v_i in $G(V, F)$. Then we consider an additional node v'_i, and we construct an edge from v_{i_j} to v'_i for each $1 \leq j \leq k$. Let $G'(V', F')$ be the network with nodes $v_1, \ldots, v_n, v'_1, \ldots, v'_n$ constructed in this way. Then, the expression pattern of the set $\{v'_1, \ldots, v'_n\}$ is determined by $v'_i = f_i(v_{i_1}, \ldots, v_{i_k})$. That is, the expression pattern of $\{v_1, \ldots, v_n\}$ corresponds to one at time t and the expression pattern of $\{v'_1, \ldots, v'_n\}$ corresponds to one at time $t+1$. Moreover, it is convenient to consider the expression pattern of $\{v_1, \ldots, v_n\}$ as the INPUT, and the expression pattern of $\{v'_1, \ldots, v'_n\}$ as the OUTPUT.

2.2 Identification Problem

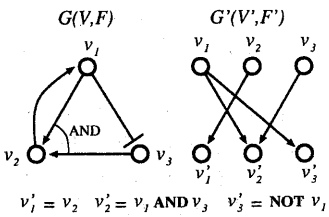
Next we review the *identification problem* according to Ref. [3]. Relating to the identification problem, we also review the *consistency problem* and the *counting problem.*

Let (I_j, O_j) be a pair of expression patterns of $\{v_1, \ldots, v_n\}$, where I_j corresponds to the INPUT and O_j corresponds to the OUTPUT. We call the pair (I_j, O_j) an *example.*

We say that a node v_i in a Boolean network $G(V, F)$ is *consistent* with an example (I_j, O_j) if $O_j(v_i) = f_i(I_j(v_{i_1}), \ldots, I_j(v_{i_k}))$ holds. We also say in such a case that $f_i(v_{i_1}, \ldots, v_{i_k})$ is consistent with an example. We say that a Boolean network $G(V, F)$ is *consistent* with (I_j, O_j) if all nodes are consistent with (I_j, O_j). For a set of examples $EX = \{(I_1, O_1), \ldots, (I_m, O_m)\}$, we say that $G(V, F)$ (resp. node v_i) is *consistent* with EX if $G(V, F)$ (resp. node v_i) is consistent with all (I_j, O_j) for $1 \leq j \leq m$. Then, the problems are defined as follows (see also Fig. 2): **CONSISTENCY:** Given n (the number of nodes) and EX, decide whether or not there exists a Boolean network consistent with EX and output one if it exists; **COUNTING:** Given n and EX, count the number of Boolean networks consistent with EX; **IDENTIFICATION:** Given n and EX, decide whether or not there exists a *unique* Boolean network consistent with EX and output it if it exists.

2.3 Simple Identification Algorithm and Sample Complexity

In this paper, we only consider the Boolean network in which the *indegree* (i.e., the number of input nodes) of each node is bounded by a constant K, because it has been proven that exponentially many examples are required in order to identify input nodes to a high indegree node [2]. The importance of indegree constraint is also pointed out in several papers [7, 10].



INPUT			OUTPUT		
v_1	v_2	v_3	v'_1	v'_2	v'_3
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

Figure 1: A Boolean network $G(V, F)$ and its wiring diagram $G'(V', F')$. In the (state transition) table, the INPUT column corresponds to the expression pattern (state) at time t and the OUTPUT column corresponds to the expression pattern (state) at time $t+1$.

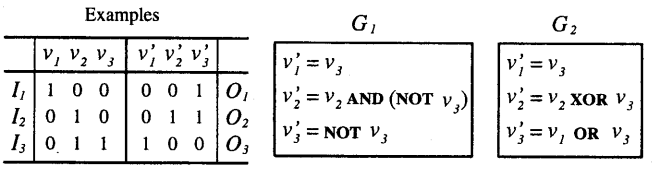


Figure 2: INPUT/OUTPUT expression patterns and Boolean networks. Boolean network G_1 is consistent with examples, while Boolean network G_2 is not consistent with the examples since node v_3 (in G_2) is not consistent with (I_3, O_3). In this case, the consistent Boolean network is not determined uniquely since we can obtain another consistent network by replacing $v_2' = v_2$ AND (NOT v_3) in G_1 with $v_2' = v_2$ XOR v_3.

Although we assume that the maximum indegree is bounded by K, all algorithms in this paper can be applied to Boolean networks whose maximum indegree is not bounded. In such a case, the algorithms correctly identify (or find) Boolean functions assigned to all nodes whose indegrees are at most K.

Here, we review a simple identification algorithm (denoted by IDENT0) proposed previously [3]. For simplicity, we describe the algorithm for the case of $K = 2$, where generalization for any K is straightforward. IDENT0 simply outputs, for each node, Boolean functions consistent with given examples in the following way: **(1)** For each node $v_a \in V$, execute step (2); **(2)** For all types of Boolean function f, for all pairs of nodes (v_j, v_k), test whether or not $O_i(v_a) = f(I_i(v_j), I_i(v_k))$ holds for all $i = 1, \ldots, m$, and output all consistent $f(v_j, v_k)$'s as possible Boolean functions assigned to v_a.

Since $O(n^2)$ pairs of nodes are examined for each v_a, IDENT0 works in $O(n^3 m)$ time [3]. Let N_a denote the number of Boolean functions produced for node v_a. Then, the consistency problem can be solved by testing whether or not $N_a > 0$ for all $a = 1, \ldots, n$, the identification problem can be solved by testing whether or not $N_a = 1$ for all $a = 1, \ldots, n$, and the counting problem can be solved by calculating $N_1 \times N_2 \times \cdots \times N_n$.

As for the number of examples required to identify the original network uniquely, the following theorem was proven.

Theorem 1.[3] If $O(2^{2K} \cdot (2K + \alpha) \cdot \log n)$ INPUT expression patterns are given uniformly at random, the following holds with probability $\geq 1 - \frac{1}{n^\alpha}$: there exists at most one Boolean network of n nodes with maximum indegree $\leq K$ which is consistent with given INPUT/OUTPUT pairs.

3 An $O(n^K m)$ Time Algorithm

In this section, we describe an improved algorithm (denoted by IDENT1) which runs in $O(n^K m)$ time. For simplicity, we only describe the algorithm for the case of $K = 2$, where generalization for any K is straightforward.

Before describing the algorithm, 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, $\overline{y}$, **AND:** $x \wedge y$, $x \wedge \overline{y}$, $\overline{x} \wedge y$, $\overline{x} \wedge \overline{y}$, **OR:** $x \vee y$, $x \vee \overline{y}$, $\overline{x} \vee y$, $\overline{x} \vee \overline{y}$, **XOR:** $x \oplus y$, $\overline{x \oplus y}$.

For simplicity, we only consider AND, OR and XOR functions here. However, other functions can be included in a straightforward way without increasing the orders of complexities. Moreover, there can be a node with indegree $> K$ in the underlying (i.e., original) Boolean network. IDENT1 still outputs correct sets of Boolean functions for nodes with indegree $\leq K$.

The idea of IDENT1 is very simple. For all Boolean functions $f(x, y)$ and for all pairs of

nodes (v_j, v_k), we generate sequences of binary numbers

$$B(f(v_j, v_k)) = \langle f(I_1(v_j), I_1(v_k)),\ f(I_2(v_j), I_2(v_k)),\ \ldots,\ f(I_m(v_j), I_m(v_k)) \rangle .$$

For each node $v_a \in V$, we let $B(v_a) = \langle O_1(v_a), O_2(v_a), \ldots, O_m(v_a) \rangle$. Then, for each node v_a, we output all $f(v_j, v_k)$'s such that $B(f(v_j, v_k)) = B(v_a)$. Of course, naive implementation would take $O(n^3 m)$ time because we compare $O(n^2)$ sequences with $O(n)$ sequences.

However, we can reduce the time complexity using the *trie*, which is a well known data structure in string matching [1] (see also Fig. 3). First, we construct a trie for $\{B(v_1), B(v_2), \ldots, B(v_n)\}$. The length (the number of edges) from the root to a node w in the trie is called the *depth* of w. For node w_d with depth d, let $r, w_1, w_2, \ldots, w_d$ be the path from the root r to w_d. Then, we let $B(w_d) = \langle label(r, w_1), label(w_1, w_2), \ldots, label(w_{d-1}, w_d) \rangle$. For each node w_d of depth d, we compute a set $CON(w_d)$ (CON means consistent) defined by

$$CON(w_d) = \{ f(v_j, v_k) |\ \langle f(I_1(v_j), I_1(v_k)), f(I_2(v_j), I_2(v_k)), \ldots, f(I_d(v_j), I_d(v_k)) \rangle = B(w_d) \} .$$

Once these $CON(w)$'s are computed for all nodes in the trie, we simply output $CON(w)$ as a set of consistent Boolean functions for each leaf w (i.e., node of depth m) because for all $v_a \in V$, there is a leaf w such that $B(w) = B(v_a)$.

Now, we analyze the time complexity. Since the length of each $B(v_i)$ is m, the trie can be constructed in $O(nm)$ time. In order to compute $CON(w)$'s efficiently, we compute $CON(w)$'s from depth i to depth $i+1$ starting with depth 0. Clearly $CON(r)$ consists of all $f(v_j, v_k)$'s. Assume that node w (of depth d) has two sons w', w'' (of depth $d+1$), where a node with single son can be processed in a similar way. Then, $CON(w') \cup CON(w'') = CON(w)$ and $CON(w') \cap CON(w'') = \emptyset$ hold. Since the length of $B(w')$ and $B(w'')$ is longer than the length of $B(w)$ by 1 (bit), each element in $CON(w)$ can be selected and copied into either $CON(w')$ or $CON(w'')$ in $O(1)$ time. Therefore, $CON(w')$ and $CON(w'')$ can be computed from $CON(w)$ in $O(|CON(w)|)$ time. Since $O(\sum_w |CON(w)|)$ is at most $O(n^2)$ where the sum is taken over all nodes of the same depth, we construct $CON(w)$'s for nodes of the same depth in $O(n^2)$ time. Since there are $m+1$ depths, the total time complexity is $O(n^2 m)$.

As for space complexity, we can reduce it to $O(nm)$ where details are omited here.

Theorem 2. The identification problem for a constant K can be solved in $O(n^K m)$ time using $O(nm)$ space.

From Theorem 1, we can see that $\sum_w |CON(w)|$ is $O(n)$ with high probability for nodes of the same depth $d \geq \Omega(\log n)$.

Corollary 1. If INPUT expression patterns are given uniformly at random, the identification problem for a constant K can be solved in $O(n^K \log n + nm)$ time on the average.

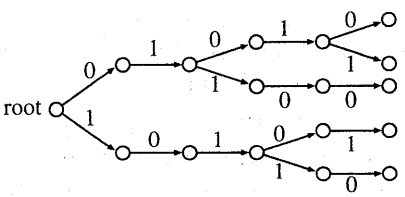


Figure 3: Example of a *trie* for binary sequences $\{01010, 01011, 01100, 10101, 10110\}$. Each edge has a label 0 or 1. Each leaf corresponds to an input binary sequence.

Note that IDENT1 can also solve the consistency problem and the counting problem in the same time and space complexities because it outputs all consistent Boolean functions of at most K inputs for each node. Moreover, IDENT1 might be extended for a generalized model of the Boolean network [3, 7] in which Boolean functions are replaced by some discrete functions.

4 An Efficient Algorithm in the Average Case

Although IDENT1 improves the time complexity, it is still high. Another drawback of IDENT1 is that it uses $O(n^K m)$ time even if we want to compute consistent Boolean functions for only a few nodes (even for only one node). Such a case might be considered when most parts of the network are already known. However, it seems difficult to develop an $o(n^2 m)$ time algorithm for $K = 2$ even if it is required to identify a set of consistent Boolean functions for only one node. Consider a case that $O_1(v_a) = O_2(v_a) = \ldots = O_m(v_a) = 0$. In order to decide only that $f(v_a) = 0$ is consistent, we must check whether or not $\{(I_i(v_j), I_i(v_k))|i = 1, \ldots, m\} = \{(0,0),(0,1),(1,0),(1,1)\}$ holds for every pair (v_j, v_k).

Although we do not completely resolve the above problems, we propose IDENT2 which works in $O(n^K m \,\mathrm{polylog}(n))$ time on the average using $O(nm)$ space, where we assume that $m = \Omega(\log n)$ INPUT patterns are given uniformly at random and OUTPUT patterns are generated from INPUT patterns according to the underlying (original) Boolean network. Moreover, if Boolean functions are restricted to AND/OR of literals, IDENT2 works in $O(n^2 m)$ time on the average (for fixed K). If Boolean functions for only a constant number of nodes are required to be found, IDENT2 works in $O(n^{K-1} m \,\mathrm{polylog}(n))$ time on the average. In this section, we first describe IDENT2 for $K = 2$ and then extend it for general K.

4.1 Identification of AND and OR

In this and the next subsections, we assume that the indegree of each node v is 1 or 2, and thus a Boolean function assigned to each node v is either UNARY, AND, OR or XOR in the underlying Boolean network. In this subsection, we describe algorithms for identifying AND and OR functions of $K = 2$, which are to be used as subroutines of IDENT2. Since IDENT2 identifies Boolean functions for each node independently, we only consider a case of identifying consistent Boolean functions for a specified node $v_a \in V$.

An algorithm for identifying AND functions is simple and is similar to the PAC learning algorithm for monotone CNF Boolean functions by Valiant [6, 12]. For simplicity, we first describe an algorithm (denoted by AND-IDENT0) for identifying Boolean functions of the form $x \wedge y$. AND-IDENT0 consists of two steps.

In the first step, AND-IDENT0 begins with the conjunction $v_1 \wedge v_2 \wedge v_3 \wedge \cdots \wedge v_n$, and processes examples (I_i, O_i) one by one. If $O_i(v_a) = 0$, nothing is done. If $O_i(v_a) = 1$, all v_j's such that $I_i(v_j) = 0$ are deleted from the conjunction. Let $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ be the nodes remaining in the conjunction after testing all examples.

In the second step, AND-IDENT0 examines all Boolean functions of $v_{i_p} \wedge v_{i_q}$ $(p < q)$, and outputs functions that are consistent with all examples.

Lemma 1. Assume that $m = \Omega(\log n)$ INPUT patterns are given uniformly at random. Then, AND-IDENT0 works in $O(nm)$ on the average for a specified node v_a.

(Proof) Since $O(n^2)$ Boolean functions are examined in the worst case and $O(m)$ time is required to examine each Boolean function, $O(n^2 m)$ time is used in the worst case. Thus, we only need to prove that AND-IDENT0 takes $O(nm)$ time with probability at least $1 - O(1/n)$.

Note that if the number of nodes remaining in the conjunction after the first step is bounded by a constant, AND-IDENT0 takes $O(nm)$ time. Therefore, we consider the probability that

the number of remaining nodes is more than $2 + C$, where C is a constant.

We first calculate the probability that the number of examples such that $O_i(v) = 1$ is less than $m/8$. Let X be the random variable denoting the number of such examples. Since the indegree of each node in the underlying network is either 1 or 2, $E(X) \geq m/4$ holds, where $E(X)$ denotes the expectation of X. From Chernoff bound [9], we obtain $Prob(X < \frac{m}{8}) < e^{-m/32}$.

Next we assume that $X \geq m/8$. From the assumption on INPUT patterns, $Prob(I_i(v_j) = 0) = 1/2$ holds for all i and for all v_j except input nodes v_p, v_q to v_a. Recall that v_j is removed if $I_i(v_j) = 0$ and $O_i(v_a) = 1$ hold for some i. Thus, the probability that each v_j ($\neq v_p, v_q$) remains after the first step is at most $(\frac{1}{2})^{\frac{m}{8}}$. Therefore, the expected number of nodes remaining in the conjunction after the first step is at most $2+(n-2)(\frac{1}{2})^{\frac{m}{8}}$. Let Y be the random variable denoting the number of remaining nodes. From Markov inequality [9], $Prob(Y - 2 > C) < \frac{(n-2)(\frac{1}{2})^{\frac{m}{8}}}{C}$. Since $e^{-m/32} + \frac{(n-2)(\frac{1}{2})^{\frac{m}{8}}}{C} < \frac{1}{n}$ holds for $m > D \log n$ for some constant D, the theorem holds. □

Modification of AND-IDENT0 for other AND functions is easy (the modified algorithm is denoted by AND-IDENT). For that purpose, we only maintain conjunctions of *literals* as in Refs.[6, 12]. An algorithm for identifying OR functions (denoted by OR-IDENT) is obtained in a similar way by using the fact that $x \vee y = \overline{\overline{x} \wedge \overline{y}}$.

4.2 Identification of XOR

For the sake of simplicity, we only describe an algorithm for identifying the input nodes for a node which is assigned the Boolean function $x \oplus y$ (denoted by XOR-IDENT0). Extension to an algorithm (denoted by XOR-IDENT) for $\overline{x \oplus y}$ functions is straightforward and omitted. The following observation plays an important role for developing and analyzing XOR-IDENT0: if $O_i(v_a) = I_i(v_j) \oplus I_i(v_k)$ holds for all i, $I_i(v_j) = \overline{I_i(v_k)}$ holds for all i such that $O_i(v_a) = 1$.

As in AND-IDENT0, XOR-IDENT0 consists of two steps: first, the set of nodes is partitioned into several classes based on the above observation; then, a procedure similar to the second step of AND-IDENT0 is applied to each class.

In the first step, as in AND-IDENT0, examples such that $O_i(v_a) = 1$ are considered. Let $i_1 < i_2 < \cdots < i_h$ be the sequence of indices of such examples. Then, the sequence of binary numbers $\langle I_{i_1}(v_j), I_{i_2}(v_j), \ldots, I_{i_h}(v_j) \rangle$ is called a *signature* of a node v_j. We say that signatures of v_j and v_k are *identical* if either $I_{i_l}(v_j) = I_{i_l}(v_k)$ holds for all $l = 1, \ldots, h$ or $I_{i_l}(v_j) = \overline{I_{i_l}(v_k)}$ holds for all $l = 1, \ldots, h$. XOR-IDENT0 partitions V into at most n classes so that each class consists of nodes with identical signatures. Let $V_1, V_2, \cdots, V_f$ be the partition.

In the second step, for each V_i, XOR-IDENT0 tests all Boolean functions of $v_j \oplus v_k$ such that $v_j, v_k \in V_i$ ($j < k$), and then outputs Boolean functions consistent with all examples.

Using a known result on the *occupancy problem* (see Section 3.1 of Ref. [9]), we have the following lemma, where the proof is omited here.

Lemma 2. Assume that $m = \Omega(\log n)$ INPUT patterns are given uniformly at random. Then, XOR-IDENT0 works in $O(nm\,(\log n / \log\log n))$ time on the average for a specified node v_a.

4.3 Integration and Extension

IDENT2 for $K = 2$ consists of identification algorithms for five categories of Boolean functions. Note that algorithms for CONSTANT functions and UNARY functions are easy and omitted here.

Theorem 3. Assume that $m = \Omega(\log n)$ INPUT patterns are given uniformly at random. Then, the consistency problem and the identification problem for $K = 2$ can be solved in $O(n^2 m\,(\log n/\log\log n))$ time on the average using $O(nm)$ space.
Moreover, all consistent Boolean functions can be found for a specified node of indegree 1 or 2 in $O(nm\,(\log n/\log\log n))$ time on the average using $O(nm)$ space.

We can extend IDENT2 for any fixed K. Here, we briefly describe the method for $K = 3$: Since $f(x,y,z) = (z \wedge f(x,y,1)) \vee (\overline{z} \wedge f(x,y,0))$ holds for any Boolean function $f(x,y,z)$, we identify $f(v_j,v_k,v_l)$ by identifying $f(v_j,v_k,1)$ using examples satisfying $I_i(v_l) = 1$, and identifying $f(v_j,v_k,0)$ using examples satisfying $I_i(v_l) = 0$.

Corollary 2. Assume that $m = \Omega(\log n)$ INPUT patterns are given uniformly at random. Then, the consistency problem and the identification problem can be solved in $O(n^K m\,\mathrm{polylog}(n))$ time on the average using $O(nm)$ space.
Moreover, if there exists at least one consistent Boolean function for a specified node, then it can be found in $O(n^{K-1} m\,\mathrm{polylog}(n))$ time on the average.

If the underlying Boolean network consists of only AND and OR nodes (of indegree at most K), we can simply extend AND-IDENT and OR-IDENT. In this case, it works in $O(n^2 m)$ time on the average using $O(nm)$ space for a constant K.

References

[1] A.V. Aho, Algorithms for finding patterns in strings, in J. Van Leeuwen (ed.) *Handbook of Theoretical Computer Science* Vol. A, 1990.

[2] T. Akutsu, S. Kuhara, O. Maruyama and S. Miyano, Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions, *Proc. 9th ACM-SIAM Symp. Discrete Algorithms*, 695–702, 1998.

[3] T. Akutsu, S. Miyano and S. Kuhara, Identification of genetic networks from a small number of gene expression patterns under the Boolean network model, *to appear in PSB'99.*

[4] A. Arkin, P. Shen and J. Ross, A test case of correlation metric construction of a reaction pathway from measurements, *Science* **277**, 1275–1279, 1997.

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

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

[7] S. Liang, S. Fuhrman and R. Somogyi, REVEAL, a general reverse engineering algorithm for inference of genetic network architectures, *Pacific Symposium on Biocomputing '98*(PSB'98), 18–29, 1998.

[8] H.H. McAdams and L. Shapiro, Circuit simulation of genetic networks, *Science* **269**, 650–656, 1995.

[9] R. Motowani and P. Raghavan, *Randomized Algorithms*, Cambridge Univ. Press, 1994.

[10] R. Somogyi and C.A. Sniegoski, Modeling the complexity of genetic networks: Understanding multigene and pleiotropic regulation, *Complexity* **1**, 45–63, 1996.

[11] R. Thomas, D. Thieffry and M. Kaufman, Dynamical behaviour of biological regulatory networks -I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state, *Bulletin of Mathematical Biology* **57**, 247–276, 1995.

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

[13] X. Wen *et. al.*, Large-scale temporal gene expression mapping of central nervous system development, *Proc. Natl. Acad. Sci. USA* **95**, 334–339, 1998.

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