

Dynamic Algorithm for Inferring Qualitative Models of Gene Regulatory Networks

Zheng Yun

BIRC, School of Comp. Eng.
Nanyang Technological University
Singapore 639798, +65-67906613
pg04325488@ntu.edu.sg

Kwoh Chee Keong

School of Comp. Eng.
Nanyang Technological University
Singapore 639798, +65-67906057
asckkwoh@ntu.edu.sg

Abstract

It is still an open problem to identify functional relations with $o(N \cdot n^k)$ time for any domain [2], where N is the number of learning instances, n is the number of genes (or variables) in the Gene Regulatory Network (GRN) models and k is the indegree of the genes. To solve the problem, we introduce a novel algorithm, DFL (Discrete Function Learning), for reconstructing qualitative models of GRNs from gene expression data in this paper. We analyze its complexity of $O(k \cdot N \cdot n^2)$ on the average and its data requirements. We also perform experiments on both synthetic and Cho et al. [7] yeast cell cycle gene expression data to validate the efficiency and prediction performance of the DFL algorithm. The experiments of synthetic Boolean networks show that the DFL algorithm is more efficient than current algorithms without loss of prediction performances. The results of yeast cell cycle gene expression data show that the DFL algorithm can identify biologically significant models with reasonable accuracy, sensitivity and high precision with respect to the literature evidences. We further introduce a method called ϵ function to deal with noises in data sets. The experimental results show that the ϵ function method is a good supplement to the DFL algorithm.

1. Introduction

With the availability of genome-wide gene expression data [12, 28], a lot of interests have been given to modelling GRNs [6, 9, 13, 14, 17, 18, 27], which are assumed to be the underlying mechanisms that regulate different gene expression patterns.

Due to the fact that very little data is available about the quantitative values of the concentrations of messenger RNA molecules and the strength of interactions between proteins and DNA, the traditional methods to simulate dynamic systems, like ordinary differential equations, can not be applied

to biological system easily. Therefore, qualitative models, like Generalized Logical Formalism (GLF) [30, 31] and Piecewise Linear Differential Equation (PLDE) [16, 22], are introduced to meet this problem.

However, it is not easy to build such GLF and PLDE models of GRNs. Currently, almost all GLF and PLDE models are built from literature [3, 11, 10, 15, 21, 24, 25]. The manual extraction of knowledge from literature clearly cumburs the applicability of these models and the speed of building them. With the recent development of microarray technology, the expression levels of thousands of genes can simultaneously be obtained at discrete time points. It is a worthy effort to make use of these data to accelerate the building of qualitative models of GRNs.

Our aim is to learn qualitative models of GRNs from discretized microarray gene expression data. The qualitative models of GRNs are a set of discrete functions which tell the regulatory relations between genes under consideration. In our method, the expression data are assumed to be the products of these functions. Then, we use a reverse engineering method based on information theory to find these functions from gene expression data.

In the identification of functional relations, it is still an open problem to develop an $o(N \cdot n^k)$ time algorithm for any domain [2]. In the following sections, we will introduce an algorithm called DFL (Discrete Function Learning) with the expected complexity of $O(k \cdot N \cdot n^2)$ to solve this open problem. The DFL algorithm is more efficient and versatile than current algorithm for reconstructing qualitative GRN models like the REVEAL algorithm [20], which is for reconstructing Boolean Networks (BLNs) from binary transition pairs, without loss of prediction performances. In addition, the DFL algorithm is automatic and requires no prior information about the regulatory relations between genes under consideration.

Gene expression data are always noisy. We further introduce a method called ϵ function to deal with the noise problems in data sets in this paper. The experimental results

show that some regulatory relations that can not be found by the DFL algorithm are successfully identified with the ϵ function method.

The rest of this paper is organized as follows. In the next section, we introduce the theory foundation of learning functional relations from data. We introduce the DFL algorithm and also analyze its complexities in section 3. We do experiments on both synthetic data sets and gene expression data of yeast cell cycle to validate the DFL algorithm in the section 4. Then, we propose a new concept called ϵ function to deal with noises in the data sets in section 5. Finally, we summarize the works of this paper in the last section.

2. Foundation of Information Theory

Our approach is based on the information theory. First of all, we introduce the following theorem, which is the theoretical foundation of our algorithm. The proof of this theorem is given in appendix A.

Theorem 2.1 *If the mutual information between \mathbf{X} and Y is equal to the entropy of Y , i.e., $I(\mathbf{X}; Y) = H(Y)$, then Y is a function of \mathbf{X} .*

From Theorem 2.1, it is straightforward to get the corollary.

Corollary 2.1 *If the joint entropy of \mathbf{X} and Y is equal to the entropy of \mathbf{X} , i.e., $H(\mathbf{X}, Y) = H(\mathbf{X})$, then Y is a function of \mathbf{X} .*

From Corollary 2.1, we see that it is sufficient to compute the joint entropy of \mathbf{X} and Y , and the entropy of \mathbf{X} to check whether Y is a function of \mathbf{X} .

3. Methods

In this section, we begin with a formal definition of the problem of reconstructing qualitative GRN models from state transition pairs. Then, we discuss how much data are sufficient to solve this problem. In the following, we introduce the DFL algorithm to solve this problem. Finally, we analyze the complexities of the DFL algorithm.

3.1. Problem definition

In qualitative models of GRNs, the genes are represented by discrete variables. The regulatory relationships between the genes are expressed by discrete functions related to each variables. Formally, a GRN $G(\mathbf{V}, \mathbf{F})$ with indegree k (the number of inputs) consists of a set $\mathbf{V} = \{X_1, \dots, X_n\}$ of nodes representing genes and a set $\mathbf{F} = \{f_1, \dots, f_n\}$ of discrete functions, where a discrete function $f_i(X_{i1}, \dots, X_{ik})$ with inputs from specified nodes X_{i1}, \dots, X_{ik} at time step

t is assigned to the node X_i at time step $t + 1$, as shown in the following equation

$$X_i(t + 1) = f_i(X_{i1}(t), \dots, X_{ik}(t)), \quad (1)$$

where $1 \leq i \leq n$.

The state of the GRN is expressed by the state vector of its nodes. We use $\mathbf{v}(t) = \{x_1, \dots, x_n\}$ to represent the state of the GRN at time t , and $\mathbf{v}(t + 1) = \{x'_1, \dots, x'_n\}$ to represent the state of the GRN at time $t + 1$. $\{x'_1, \dots, x'_n\}$ is calculated from $\{x_1, \dots, x_n\}$ with Equation 1. A state transition pair is $\mathbf{v}(t) \rightarrow \mathbf{v}(t + 1)$.

When using BLNs to model GRNs, genes are represented with binary variables with two values ON (1) and OFF (0), which means the genes are turned on or turned off respectively. In addition, the f_i s in Equation 1 are Boolean functions. When using GLFs and PLDEs to simulate GRNs, the f_i s in Equation 1 are multi-value discrete functions.

The problem of inferring the qualitative model of the GRN from input-output transition pairs (time series of gene expression) is defined as follows.

Definition 3.1 *Let $\mathbf{V} = \{X_1, \dots, X_n\}$. Given a transition table $T = \{\mathbf{v}(t) \rightarrow \mathbf{v}(t + 1)\}$ where $\mathbf{v}(t)$ is the state vector of the GRN model at time t , find a set of discrete functions $\mathbf{F} = \{f_1, f_2, \dots, f_n\}$, so that $X_i(t + 1)$ (X'_i hereafter) is calculated from f_i as follows*

$$X_i(t + 1) = f_i(X_{i1}(t), \dots, X_{ik}(t)), \quad (2)$$

where t goes from 1 to a limited constant N .

If the \mathbf{F} are Boolean functions, then the GRN model is a BLN, otherwise the GRN model is a GLF or PLDE model.

3.2. Data quantity

We discuss how much data is necessary to successfully infer \mathbf{F} in this section. Akutsu *et al.* [1] proved that $\Omega(2^k + k \log_2 n)$ transition pairs are the theoretic lower bound to infer the BLNs, where n is the number of genes and k is the maximum indegree of these genes.

Theorem 3.1 (Akutsu 1998) *$\Omega(2^k + k \log_2 n)$ transition pairs are necessary in the worst case to identify the Boolean network of maximum indegree $\leq k$.*

To meet the requirement of multi-state discrete functions in GLF and PLDE models, we introduce the following theorem, which is a generalization of Theorem 3.1. The proof of the theorem is also given in appendix A.

Theorem 3.2 *$\Omega(b^k + k \log_b n)$ transition pairs are necessary in the worst case to identify the qualitative GRN models of maximum indegree $\leq k$ and the maximum number of discrete level for variables $\leq b$.*

Hereafter, we use “ b (base)” to denote the number of discrete level for variables.

In the DFL algorithm, we introduce a coefficient c to determine the actual size of synthetic data sets as follows,

$$N = c \times (b^k + k \log_b n). \quad (3)$$

That is to say, the parameter t in Definition 3.1 goes from 1 to N .

3.3. Search method

From Theorem 2.1, the problem in Definition 3.1 is converted to finding a set of input genes whose mutual information with Y' is equal to the entropy of Y' for each gene Y in the GRN.

For n discrete variables $\mathbf{V} = \{X_1, \dots, X_n\}$, there are totally 2^n subsets. Clearly, it is NP-hard to examine all subsets of \mathbf{V} exhaustively. However, for GRNs, each gene is estimated on the average to interact with four to eight other genes [4]. Therefore, by restricting the indegree of a gene to a limited integer k , the problem can be solved in polynomial time. Even when we do this compromise, it is still very difficult to solve the problem. As mentioned before, it is still an open problem to develop an $o(N \cdot n^k)$ time algorithm for identify functional relations in any domain [2]. In the following, we will introduce an algorithm called DFL (Discrete Function Learning) with the complexity of $O(k \cdot N \cdot n^2)$ to solve this open problem. The main steps of the DFL algorithm are listed in Table 2. In the DFL algorithm, we use the following definition, called Δ supersets.

Definition 3.2 Let \mathbf{X} be a subset of $\mathbf{V} = \{X_1, \dots, X_n\}$, then $\Delta_i(\mathbf{X})$ of \mathbf{X} are the supersets of \mathbf{X} so that $\mathbf{X} \subset \Delta_i(\mathbf{X})$ and $|\Delta_i| = |\mathbf{X}| + i$, where $|\mathbf{X}|$ denotes the cardinality of \mathbf{X} .

To clarify the heuristic underlying the DFL algorithm, let us consider a BLN consisting of four genes, as shown in Figure 1. In this example, the function of each gene is listed in Table 1. The set of all genes is $\mathbf{V} = \{A, B, C, D\}$, and we use \mathbf{X} to denote subsets of \mathbf{V} .

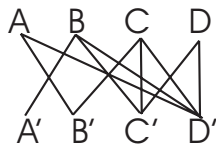


Figure 1. The wiring diagram of a BLN model, where $n = 4$, $k_{max} = 4$. X' denotes the state of X in next time step.

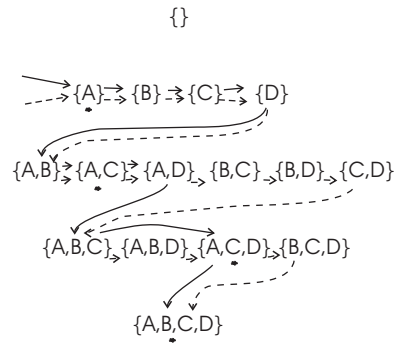


Figure 2. Search procedures of the DFL algorithm and the REVEAL algorithm when finding the Boolean function of D' in Figure 1. The solid line is for the DFL algorithm, the dashed line is for the REVEAL algorithm. The combinations with a black dot under them are the subsets which share the largest mutual information with D' on their layers. The REVEAL algorithm firstly searches the first layer (the subsets with one gene), then the second layer, and so on. Finally, it finds the target subsets $\{A, B, C, D\}$ at the fourth layer. The DFL algorithm uses a different heuristics. Firstly, it searches the first layer, then finds that $\{A\}$, with a black dot under it, shares the largest mutual information between D' among subsets on the first layer. Then, it continues to search $\Delta_1(A)$ on the second layer. Similarly, these calculations continue until the target combination $\{A, B, C, D\}$ is found on the fourth layer.

One of the commonly used algorithms to infer BLNs from data is the REVEAL algorithm [20]. As shown in Figure 2, the REVEAL algorithm uses an exhaustive search method, it first searches the subsets with only one gene, then subsets with two genes, and so on. When compared with the REVEAL algorithm, the DFL algorithm uses a better heuristic when finding the target combination. Firstly, the DFL algorithm searches the first layer, then it sorts all subsets on the first layer. It finds that $\{A\}$ shares the largest mutual information with D' among subsets on the first layer. Then, the DFL algorithm searches through $\Delta_1(A)$, \dots , $\Delta_{k-1}(A)$, however it always decides the search order of $\Delta_{i+1}(A)$ bases on the calculation results of $\Delta_i(A)$.

The DFL algorithm guarantees the check of every subset whose cardinality is not larger than k . There are $\sum_{i=1}^k \binom{n}{i} \approx n^k$ subsets whose cardinalities are smaller than or equal to k . In addition, the length N of transition table T is $O(b^k + k \log_b n)$. Since there are n

Gene	Rule
A	$A' = B$
B	$B' = A + C$
C	$C' = (B \cdot C) + (C \cdot D) + (B \cdot D)$
D	$D' = (A \cdot B) + (C \cdot D)$

Table 1. Boolean functions of the example, where “+” is the logical OR operation, and “.” is the logical AND operation.

Algorithm: DFL(V, k, T)

Input: V with n genes, indegree k ,
 $T = \{v(t) \rightarrow v(t+1)\}, t = 1, \dots, N$.

Output: $F = \{f_1, f_2, \dots, f_n\}$

Begin:

```

1  L ← all single element subsets of V;
2  ΔTree.FirstNode ← L;
3  for every gene Y ∈ V {
4    calculate H(Y'); //from T
5    D ← 1; //initial depth
6*  F.add(Sub(Y, ΔTree, H(Y'), D, k));
  }
7  return F;
End

```

* The *Sub()* is a sub routine listed in Table 3.

Table 2. The DFL algorithm.

genes in the network, the complexity of the DFL algorithm is $O((b^k + k \log_b n)n^{k+1})$ in the worst case. Contributing to sort step in the line 7 of the sub routine, the algorithm makes the best choice in current layer of subsets. Since there are $(n-1)$ Δ_1 supersets for a given single element subset, $(n-2)$ Δ_1 supersets for a given two element subsets, and so on. The DFL algorithm only considers $\sum_{i=0}^{k-1} (n-i) \approx kn$ subsets on the average. Thus, the time complexity of the DFL algorithm is approximately $O(k \cdot n^2 \cdot (N + \log n))$ on the average, where $\log n$ is for sort step in line 7 of Table 3 and N is for the length of input table T . By ignoring the minor terms, the time complexity of the DFL algorithm becomes $O((kb^k + k^2 \log_b n)n^2)$ on the average. The expected complexity of the DFL algorithm comes from three parameters k , b and n . The complexity $O((kb^k + k^2 \log_b n)n^2)$ grows quasi-squarely with n when $k, b \ll n$, multinomially with b , and exponentially with k .

To store the information needed in the search processes, the DFL algorithm uses two data structures. The first data structure used by the DFL algorithm is a linked list, which stores the state table of every gene during its calculation

Algorithm: *Sub*($Y, \Delta Tree, H, D, k$)

Input: $Y, \Delta Tree$, entropy $H(Y)$

current depth D , indegree k

Output: function $Y = f(\mathbf{X})$

Begin:

```

1  L ← ΔTree.DthNode;
2  for every element X ∈ L {
3    calculate I(X, Y);
4    if(I(X, Y) == H) {
5*   extract Y = f(X) from T;
6    return Y = f(X);
  }
  }
7  sort L according to I;
8  for every element X ∈ L {
9    if(D < k){
10   D ← D + 1;
11   ΔTree.DthNode ← Δ1(X);
12   return Sub(Y, ΔTree, H, D, k);
  }
  }
13 return "Fail(Y)";
End

```

* By deleting unrelated variables and duplicate rows in T .

Table 3. The sub routine of the DFL algorithm.

process. Every gene has two sequences representing its state of current time step and next time step respectively. According to Equation 3, the space complexity of the first data structure is $O((b^k + k \log_b n)n)$.

The second one is a two-dimension linked list called $\Delta Tree$ of length k where each node in the first dimension is itself a linked list. This data structure is used to store the Δ supersets in the calculation procedures. More precisely, the first node of this data structure is used to store the single element subsets. If the DFL algorithm is processing $\{X_i\}$ and its Δ supersets, the second node to the k th node are used to store Δ_1 to Δ_{k-1} supersets of $\{X_i\}$. If there are n genes, there would be $\sum_{i=0}^{k-1} (n-i) \approx kn$ subsets in the $\Delta Tree$. To store the $\Delta Tree$, the space complexity would be $O(kn)$, since only the indexes of the genes are stored for each subsets. Therefore, the total space complexity of the DFL algorithm is $O((b^k + k \log_b n)n)$.

4. Results

In this section, we first introduce the synthetic data sets of BLN models that we use. Then, we discuss two kinds of

1 Except Δ_1 supersets, only a part of other $\Delta_i (i = 2, \dots, k-1)$ supersets is stored in $\Delta Tree$.

experiments of BLNs to validate the efficiency of the DFL algorithm in comparison with the REVEAL algorithm. In the following, we conduct experiments on synthetic data sets to show the sensitivity of the DFL algorithm. Next, we perform experiments on synthetic data set of a GLF model. Finally, we do experiments on yeast cell cycle gene expression profile of Cho *et al.* [7].

We implement the DFL algorithm and the REVEAL algorithm with the Java language version 1.4.1. The implementations are included in our software called Java Gene Networks. We perform our experiments on an HP *AlphaServer* SC computer, with one EV68 1GHz CPU and 1GB memory, running *Tru64* Unix operating system.

4.1. Synthetic data sets of BLNs

We discuss synthetic data sets of BLNs in this section.

For a BLN consisting of n genes, the total state space would be 2^n . In our implementation, we generate the left sides of the transition pairs according to Discrete Uniform Distribution, i.e., $p(i) = \frac{1}{2^n}$, where i randomly choose one value from 0 to $2^n - 1$ inclusively.

For simplicity, we assume the Boolean functions between genes are all **OR** operation, although other functions are also feasible. There are $\binom{n}{k}$ subsets with cardinality of k , which form the k th layer in the searching process as shown in Figure 2. The different position of the target combination in this layer will lead to very different run time for the REVEAL algorithm. To average the different run time caused by different position of combinations, we use two different data sets for the same n, k . In one kind of data sets, all genes are determined by the first k elements of gene list \mathbf{V} . We name these data sets “head”. In the other kind of data sets, all genes are determined by the last k genes of \mathbf{V} , we name them “tail”.

4.2. Experiments when k is fixed, n increases

In this section the indegree of each gene k is fixed to 3, and the number of transition pairs is calculated with Equation 3 where c is 3. The number of genes goes from 20 to 100.

The experiment results are shown in Figure 3 (a), where the time is the average value of 5 experiments on “head” data sets and 5 experiments on “tail” data sets. The run time values are shown in logarithmic value. In all experiments of this kind, both the DFL algorithm and the REVEAL algorithm can find the original BLNs correctly. However, the DFL algorithm is significantly faster than the REVEAL algorithm as shown in Figure 3 (a).

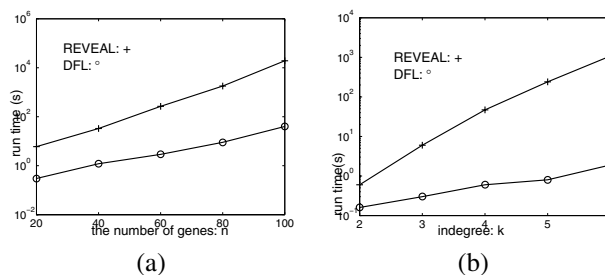


Figure 3. Run time of the DFL algorithm and the REVEAL algorithm. (a) when n increases alone, (b) when k increases alone.

4.3. Experiments when n is fixed, k increases

In this section, the number of genes n is fixed to 20, and k is increased from 2 to 6. Similar to the results of the prior section, both the DFL algorithm and the REVEAL algorithm can find the original BLNs correctly. However, the search times are also significantly different as shown in Figure 3 (b), where the time is the average value of 5 experiments on “head” data sets and 5 experiments on “tail” data sets. The run time values are also shown in logarithmic value.

As shown in the time complexity of the DFL algorithm $O((kb^k + k^2 \log_b n)n^2)$, it will grow in an exponential way with k . In Figure 3 (b), the run time of the DFL algorithm grows approximately linearly in logarithmic coordinate, which means an exponential growth in ordinary coordinate.

Also, as indicated by Figure 3 (b), the run times of the DFL algorithm are significantly smaller than those of the REVEAL algorithm in all cases.

4.4. Experiments for large n

We do experiments that the number of genes n goes from 1000 to 6000, which is approximately the number of genes with in a yeast genome, in this section. As discussed in the section 3, each gene is estimated on average to interact with four to eight other genes [4]. Another kind of experiments for k are also done, where n is fixed to 1000 and k goes from 2 to 8. Here, we do not do experiments for the REVEAL algorithm, which has already been inoperative *per se*.

Again, the DFL algorithm can correctly identify the original BLNs in all experiments. From Figure 4 (a), we see the run time of DFL grows quasi-squarely with n . In Figure 4 (b), the run time of the DFL algorithm grows approximately linearly in logarithmic coordinate with k , which means an exponential growth in ordinary coordinate.

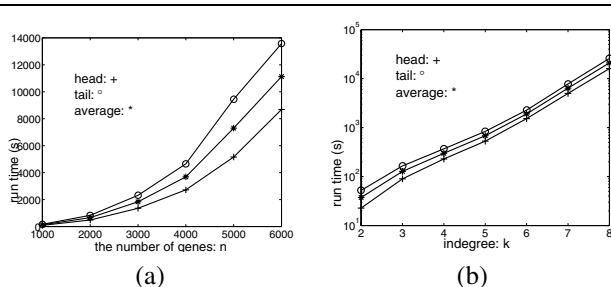


Figure 4. The run time of the DFL algorithm. (a) when n increases alone, where $k = 3, c = 4$, (b) when k increases alone, where $n = 1000$.

4.5. Experiments of sensitivity

In this section, we do some experiments to show the sensitivity of the DFL algorithm. We change the number of learning instances (N) in this kind of experiments, since when the data is not enough the algorithm may fail to identify the original BLNs. The results are shown in Figure 5, where we do experiments for BLNs of $n = 50$ and $n = 100$, and each point is the average value of five “head” and five “tail” data sets.

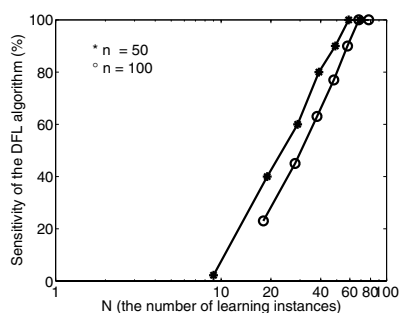


Figure 5. The sensitivity of the DFL algorithm. The horizontal axis, N (the number of learning instances), is log-scaled. The right most points of the two curves correspond to the numbers of learning instances obtained by $c = 3$ of Equation 3

Sensitivity measures the percentage of correct positive predictions by the DFL algorithm. From Figure 5, it is shown that the sensitivity of the DFL algorithm grows linearly with the logarithmic value of the number of learning instances. However, the sensitivity will become one and not increase after the number of learning instances grows to a

certain number. That means, if the data is enough, the DFL algorithm can correctly identify the original BLNs.

4.6. Experiment on data of a GLF model

In this section, we use the DFL algorithm to find a GLF model discussed in [29] and shown in Figure 6.

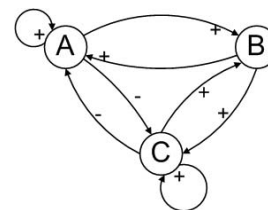


Figure 6. A simple GLF model of GRN. The genes are represented by circles. The directed edges represents the regulatory relations between genes. The “+” and “-” beside an edge represents this regulatory relation is activation and repression respectively.

In [29], the authors provided the transition table of the model, which consists of 18 lines. We use this transition table as the input table T of the DFL algorithm. The DFL algorithm can correctly find that $A' = f_1(A, B, C)$, $B' = f_2(A, C)$, $C' = f_3(A, B, C)$. The learned f_i s are truth tables, since we still lack the tools to simplify the multi-value discrete functions like the Kaunaugh maps for Boolean functions. In addition, the activation or repression relations in the graph could be obtained by analyzing the correlation coefficient between genes [5]. The correlation coefficient matrix of the example in Figure 6 is listed in Table 4.

	A'	B'	C'
A	0.2	0.6	-0.7
B	0.3	0	0.3
C	-0.7	0.6	0.2

Table 4. The correlation coefficient matrix of the GLF example in Figure 6.

In the correlation coefficient matrix, positive, negative and zero values indicate activation, repression and no direct interaction respectively. In our example, the 0.2 in the first column of the first line in Table 4 means A gives activation to A' , and so on. We see that the activation and re-

pression relations in Figure 6 are correctly identified with the correlation coefficient matrix in Table 4.

4.7. Experiments on yeast gene expression data

In this section, we use the gene expression data of yeast *Saccharomyces cerevisiae* cell cycle from Cho *et al.* [7], which covers approximately two full cell cycles [7]. In [19], Lee *et al.* reported a GRN related to cell cycle of yeast. The GRN consists of 11 well-known yeast cell cycle regulators, which are Mbp1, Swi4, Swi6, Mcm1, Fkh1, Fkh2, Ndd1, Swi5, Ace2, Skn7 and Stb1. The Mcm1, Swi5, Ace2 and Stb1 are relatively loosely related to other genes. Thus, we only consider the remaining 7 genes. We discretize the data set in [7] to three and four levels, then rearrange these expression values to state-transition pairs such that the expression values at current time step are the product of expression values at the prior time step. Finally, we apply the DFL algorithm on the obtained transition table. The learned models are shown in Figure 7.

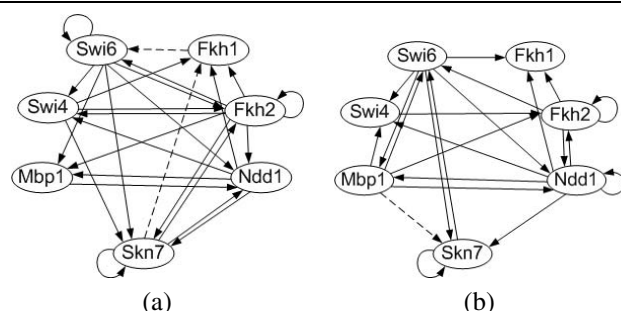


Figure 7. The learned GRN model. (a) The number of discrete levels for gene expression value is 3 and the indegree of the GRN is set to 5. (b) Idem, where the base for gene expression value is 4. The regulators are represented by ovals. The directed edge from Gene A to Gene B means that Gene A is a regulator of Gene B. The solid edges represent regulatory relations that have been verified by other approaches. The dashed edges represent regulatory relations that have not been verified.

The DFL algorithm is automatic and requires no prior knowledge of the regulatory relations between the genes under consideration. The DFL algorithm is also quite efficient, only needs less than 0.2 seconds for all experiments done.

The literature evidence for regulatory relations represented in Figure 7 are shown in Table 5. For in-

stance, Swi4 transcription is regulated in late G1 by both SBF(Swi4/Swi6) and MBF(Mbp1/Swi6) [26]. In Figure 7, these regulatory relations are identified in (a) and (b) respectively.

Gene	Regulator (Protein)							Evidence
	M1	S4	S6	F1	F2	N1	S7	
MBP1	*3	*	*34		*34	*34		[19], [26]
SWI4	*34	*3	*34		*3	*34	*	[19], [26]
SWI6	*4	*	*34	3	*34	*	*4	[19], [26]
FKH1	*4	*3	*4	*	*34	*34	3	[19], [26]
FKH2	*4	*34	*3	*3	*34	*4	*3	[19], [26]
NDD1	*34	*	*34	*	*34	*4	*3	[19], [26]
SKN7	34	*3	*34		*3	*34	*34	[19]

“*” means regulatory relations. For example, “*” in the first cell of first line means that Mbp1 gives MBP1 gene autoregulation [19]. “3” and “4” represent the regulatory relations found with the DFL algorithm when the bases for expression values are 3 and 4 respectively. M1, S4, S6, F1, F2, N1 and S7 are Mbp1, Swi4, Swi6, Fkh1, Fkh2, Ndd1 and Skn7 respectively.

Table 5. The literature evidences for the GRN model in Figure 7 and Figure 8.

From Table 5, we obtain the accuracy, sensitivity and precision of the DFL algorithm, and tabulate them in Table 6. In Table 6, we see that approximate 83 percent of the regulatory relations which have literature evidences are found with the DFL algorithm, when we combine the results from both Figure 7 and Figure 8. It is also shown that the precision of the DFL algorithm is quite high no matter what the bases for expression values are. That means, it is quite probable that the regulatory relations found with the DFL algorithm are biologically meaningful. In Table 6, it is shown that over 90 percent of the regulatory relations found by the DFL algorithm are biologically significant.

	Accur.	Sensi.	Preci.
DFL ($b = 3$)	65	67	90
DFL ($b = 4$)	63	60	96
DFL (Combined)	80	83	92
K2 ($b = 3$)	27	17	88
K2 ($b = 4$)	22	12	83
K2 (Combined)	33	24	91

Table 6. The accuracy, sensitivity and precision of the DFL algorithm and the K2 algorithm.

To do a comparison with another commonly used model, Bayesian networks, we apply the K2 algorithm [8] on the same data sets. Then, we calculate the accuracy, sensitiv-

ity and precision of the K2 algorithm with respect to literature evidences, and list them in Table 6 also. In Table 6, it is shown that the measures of the K2 algorithm are substantially lower than those of the DFL algorithm. Another important thing is that the autoregulations can not be represented by Bayesian networks due to fact that the structures of them are directed acyclic graphs [23]. Therefore, the autoregulations are predeterminedly missed whatever algorithms for learning Bayesian networks are used. However, the autoregulations are very common in GRNs as shown in Table 5, in which the diagonal line from upper-left corner to lower-right corner is fully occupied with autoregulation evidences.

Some regulatory relations which have literature evidence are not found by the DFL algorithm, as shown in Table 5. This is also shown by the sensitivity value in Table 6. There are mainly two reasons for this discrepancy. First, the size of the data set is too small. Second, there are noise in the gene expression data. It is reasonable to expect that the model obtained from the DFL algorithm will become more reasonable when the input data is larger and more precise. Further, there are also some regulatory relations (represented by dashed edges) to be verified yet. When we calculate the measures in Table 6, we count these relations as false positives.

5. ϵ Function

Due the the noises in the gene expression data, the requirement of Theorem 2.1 may not be satisfied strictly. Therefore, some regulatory relations can not be identified successfully. In this section, we introduce the concept of ϵ function to meet issues incurred by noises in the data sets.

5.1. The definition of ϵ function

When there are noises in the data sets, the requirement of Theorem 2.1 can not be satisfied strictly. In these cases, we can relax the requirement to obtain a compromised result. Formally, we define the ϵ function as follows.

Definition 5.1 *If $H(Y) - I(\mathbf{X}; Y) \leq \epsilon \times H(Y)$, then $Y = f_{\epsilon}(\mathbf{X})$ where ϵ is a significant factor.*

Correspondingly, the line 4 of Table 3 should be modified.

5.2. ϵ function for gene expression data

We do experiments on yeast cell cycle data from Cho *et al.* [7] too. The results in shown in Figure 8.

As shown in Figure 8 (a) and (b), some regulatory relations that are not found in 7 (a) are identified with the ϵ function method. For example, the autoregulation of Mph1 and Swi4 are successfully found in Figure 8 (a). In Figure

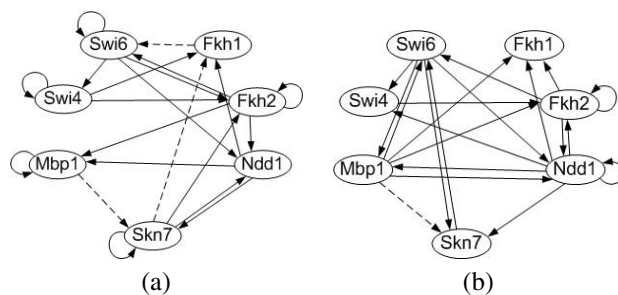


Figure 8. The learned GRN model for yeast cell cycle with the ϵ function method. (a) The base for gene expression value is 3, the indegree of the GRN is 5, and the ϵ is 0.2. (b) The base for gene expression value is 4, the indegree of the GRN is 5, and the ϵ is 0.15. The legends are the same as those of Figure 7.

8 (b), the regulation of Fkh1 by Mbp1 is identified. In addition, the regulation of Fkh2 by Fkh1 is identified in experiments of $b = 3$ and $\epsilon = 0.25$ (not shown in Figure 8).

However, some regulatory relations also disappear when we do compromise in the ϵ function method. For instance, the regulation of Fkh1 by Fkh2 disappears in Figure 8 (a). Generally, the GRN model tends to become scarcer (contain fewer edges) when the value of ϵ increases. This is due to the fact that fewer genes can satisfy the requirement of ϵ function when the value of ϵ increases.

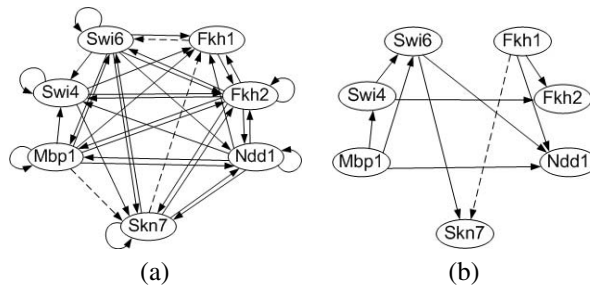


Figure 9. The combined GRN models. (a) Combined model of Figure 7 and Figure 8. (b) Combined Bayesian network structure learned with the K2 algorithm where the base for expression value is set to 3 and 4 respectively. The legends are the same as those of those of Figure 7.

Finally, we give unified models in Figure 9, in which (a) combines results in both Figure 7 and Figure 8, and (b) is

a combined Bayesian network model learned by the K2 algorithm when the base for expression value is 3 and 4. It is shown in Figure 9 that the model found with the DFL algorithm is more significant than that learned with the K2 algorithm. As we mentioned before, there are no autoregulations found in Figure 9 (b), but 6 out of 7 autoregulations are found in Figure 9 (a).

6. Conclusions

The contributions of this paper are three fold.

First, we systematically analyze a way to find functional relations from an information theory approach. That is if the mutual information between \mathbf{X} and Y is equal to the entropy of Y , then Y is a function of \mathbf{X} .

Second, we introduce a new algorithm, called DFL (Discrete Function Learning), to learn qualitative models of GRNs from microarray gene expression data. The DFL algorithm is a general method to find discrete functional relations. The excellence of the DFL algorithm consists in that the base for gene expression data is adjustable. This virtue makes it possible to find GRN models of binary values (BLNs) and multi-state values (GLF *etc.*) with a universal tool. The experimental results show that it can correctly find the original model of the synthetic data set, and identify biologically significant models from a very limited gene expression data set. In addition, we analyze the theoretic lower bound of the size of data sets to accomplish the task of finding these discrete functions. The DFL algorithm is superior to currently existing algorithms with the time complexity of $O((kb^k + k^2 \log_b n)n^2)$ on the average, although its worst case complexity is $O((b^k + k \log_b n)n^{k+1})$. We also do experiments on synthetic data sets to validate our analysis about the complexity of the DFL algorithm. In our experiments, we also find that the sensitivity of the DFL algorithm grows linearly with the logarithmic value of the number of learning instances.

At last, we introduce a new concept called ϵ function to deal with the noises in data sets. The experiments on yeast cell cycle expression data show that the ϵ function method is a good supplement to the DFL algorithm.

In the future, there are at least two ways to extend the DFL algorithm. First, it is advisable to incorporate other kinds of data, like genome-wide location data, in the learning procedures of qualitative models. Second, we can automatically explore whether a regulator is an activator or a repressor by calculating the correlation between the regulator and the regulated gene as shown in section 4.6.

ACKNOWLEDGEMENTS

The authors appreciate Prasanna R Kolatkar and Ng See-Kiong for their reviews on an early version of this paper.

References

- [1] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Proceedings of Pacific Symposium on Biocomputing '99*, volume 4, pages 17–28, 1999.
- [2] T. Akutsu, S. Miyano, and S. Kuhara. Algorithm for identifying boolean networks and related biological networks based on matrix multiplication and fingerprint function. *Journal of Computational Biology*, 7(3/4):331–343, 2000.
- [3] R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. *Lecture Notes in Computer Science*, 2034:19–32, 2001.
- [4] M. Arnone and E. Davidson. The hardwiring of development: organization and function of genomic regulatory systems. *Development*, 124:1851–1864, 1997.
- [5] Z. Bar-Joseph, G. K. Gerber, T. I. Lee, N. J. Rinaldi, J. Y. Yoo, F. Robert, D. B. Gordon, E. Fraenkel, T. S. Jaakkola, R. A. Young, and D. K. Gifford. Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, 21:1337–1342, 2003.
- [6] H. Bolouri and E. Davidson. Modeling transcriptional regulatory networks. *BioEssays*, 24:1119–1129, 2002.
- [7] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73, 1998.
- [8] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309, 1992.
- [9] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.
- [10] H. de Jong, J. Geiselman, G. Batt, C. Hernandez, and M. Page. Qualitative simulation of the initiation of sporulation in *B. subtilis*. Technical Report 4527, INRIA, 2002.
- [11] H. de Jong, M. Page, C. Hernandez, and J. Geiselman. Qualitative simulation of genetic regulatory networks: Method and application. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 67–73, San Mateo, CA, 2001. Morgan Kaufmann.
- [12] J. DeRisi, V. Iyer, and P. Brown. Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale. *Science*, 278(5338):680–686, 1997.
- [13] P. D'haeseleer, S. Liang, and R. Somogyi. Genetic networks inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.
- [14] D. Endy and R. Brent. Modelling cellular behavior. *Nature*, 409(6818):391–395, 2001.
- [15] R. Ghosh and C. J. Tomlin. Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. *Lecture Notes in Computer Science*, 2034:232–246, 2001.

- [16] L. Glass and S. Kauffman. The logical analysis of continuous non-linear biochemical control networks. *Journal of Theoretical Biology*, 39:103–129, 1973.
- [17] J. Hastly, D. McMillen, and J. Collins. Engineered gene circuits. *Nature*, 420:224–230, 2002.
- [18] J. Hastly, D. McMillen, F. Isaacs, and J. Collins. Computational studies of gene regulatory networks: in numero molecular biology. *Nature Review Genetics*, 2(4):268–279, 2001.
- [19] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional Regulatory Networks in *Saccharomyces cerevisiae*. *Science*, 298(5594):799–804, 2002.
- [20] S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithms for genetic network architectures. In *Proceedings of Pacific Symposium on Biocomputing '98*, volume 3, pages 18–29, 1998.
- [21] L. Mendoza, D. Thieffry, and E. Alvarez-Buylla. Genetic control of flower morphogenesis in *Arabidopsis thaliana*: A logical analysis. *Bioinformatics*, 15(7-8):593–606, 1999.
- [22] T. Mestl, E. Plahte, and S. Omholt. A mathematical framework for describing and analysing gene regulatory networks. *Journal of Theoretical Biology*, 176:291–300, 1995.
- [23] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.
- [24] L. Sanchez and D. Thieffry. A logical analysis of the *Drosophila* gap genes. *Journal of Theoretical Biology*, 211:115–141, 2001.
- [25] L. Sanchez, J. van Helden, and D. Thieffry. Establishment of the dorso-ventral pattern during embryonic development of *Drosophila melanogaster*: A logical analysis. *Journal of Theoretical Biology*, 189:377–389, 1997.
- [26] I. Simon, J. Barnett, N. Hannett, C. Harbison, T. Rinaldi, N.J. and Volkert, J. Wyrick, J. Zeitlinger, D. Gifford, T. Jaakkola, and R. Young. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 166:679–708, 2001.
- [27] P. Smolen, D. Baxter, and J. Byrne. Modeling transcriptional control in gene network: Methods, recent results, and future directions. *Bulletin of Mathematical Biology*, 62:274–292, 2000.
- [28] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.
- [29] D. Thieffry and R. Thomas. Qualitative analysis of gene networks. In *Proceedings of Pacific Symposium on Biocomputing '98*, volume 3, pages 77–88, 1998.
- [30] R. Thomas and R. d’Ari. *Biological Feedback*. CRC Press, Boca Raton, FL, 1990.
- [31] 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(2):247–276, 1995.
- [32] R. W. Yeung. *A First Course in Information Theory*. Kluwer Academic/Plenum Publishers, New York, NY, 2002.

A. The Proofs

Yeung [32] gave a proof for the following theorem.

Theorem A.1 $H(Y|X) = 0$ if and only if Y is a function of X .

Since $I(\mathbf{X}; Y) = H(Y) - H(Y|\mathbf{X})$ and $H(Y|\mathbf{X}) = H(\mathbf{X}, Y) - H(\mathbf{X})$, directly from Theorem A.1, it is straightforward to obtain Theorem 2.1 and Corollary 2.1.

The proof for Theorem 3.2 is given as follows.

Theorem A.2 $\Omega(b^k + k \log_b n)$ transition pairs are necessary in the worst case to identify the qualitative GRN models of maximum indegree $\leq k$ and the maximum base for variables $\leq b$.

Proof. Firstly, we consider the number of mutually distinct qualitative GRN models of maximum indegree $\leq k$ and the maximum base for variables $\leq b$.

There are $\binom{n}{k} \approx n^k$ possible combinations of inputs for a given gene, and b^{b^k} possible possible discrete functions of base b for each gene. Thus, there are $\Omega((b^{b^k} \cdot n^k)^n)$ qualitative GRN models of maximum indegree $\leq k$ and the maximum base for variables $\leq b$.

Therefore, $\Omega(b^k \cdot n \log_2 b + nk \log_2 n)$ bits are required to represent a GRN model of maximum indegree $\leq k$ and of maximum indegree $\leq k$ and the maximum base for variables $\leq b$.

At last, we consider the number of transition pairs. For each transition pair, the information quantity is $n \log_2 b$ bits if the maximum base for variables $\leq b$. Hence, $\Omega(b^k + k \log_b n)$ transition pairs are required in the worst case.