

EFFICIENT SYNTHESIS OF A CLASS OF BOOLEAN PROGRAMS FROM I-O DATA: APPLICATION TO GENETIC NETWORKS

RUTH CHARNEY, JACQUES COHEN, AND AURÉLIEN RIZK

ABSTRACT. The paper addresses the problem of synthesizing programs using a restricted set of input-output data. The program to be synthesized consists of assigning to individual Boolean variables x' , the result of evaluating functions involving other Boolean variables x . The program can be initially viewed as a black box having n inputs corresponding to the variables x and n outputs corresponding to the x' . The goal is to determine the Boolean functions that produce x' given x . It is shown that by suitably selecting a limited number of input sets to the black box and examining their output, the assignments can be fully reconstructed. The paper describes an effective representation of the Boolean functions, identifies related work, and provides an upper bound on the number of *I-O* pairs needed to rebuild the program. The work is based on discrete Jacobians that are computed from the selected *I-O* pairs. Finally, it is shown that the synthesis can be extended to programs whose variables are bounded integers. The results indicate that, if each Boolean function involves a small number of variables, a limited set of selectively chosen *I-O* pairs suffices to synthesize the program (instead of a potentially exponential number of pairs that would be needed in a worst case scenario). These results are of interest in the reverse engineering of genetic networks from biological experiments.

1. INTRODUCTION

We consider the class of Boolean programs P with n pairs of Boolean variables \mathbf{x} and \mathbf{x}' ; P consists of n assignments of the type $x'_i \leftarrow f_i(\mathbf{x})$ where f_i is any Boolean function involving the variables \mathbf{x} and the variables \mathbf{x}' store the new values obtained by evaluating the f_i 's. Given an input vector I standing for the initial values for the variables x_i , the above assignments compute an output vector O displaying the new values for the Boolean variables x'_i . The problem that interests us is: *Given a limited set of I-O data, synthesize P .*

In many cases, f_i is independent of some of the variables in \mathbf{x} and we call these variables *fictitious* or non-essential variables for f_i . For example, if $\mathbf{x} = (x_1, x_2)$, but $f_i(\mathbf{x}) = \text{not } x_1$, then x_2 is fictitious, and f_i may be considered a function of the single variable x_1 .

R. Charney was partially supported by NSF grant DMS 0705396.

The program P can be viewed as a directed graph G in which the variables \mathbf{x} represent the nodes, which are additionally labeled by the f_i 's. The incoming edges to the i th node are those emanating from the nodes represented by the essential variables for f_i . This is called the *in-degree* of the node. The maximum number of such edges is called the maximum in-degree of G . The graphs G , usually referred to as genetic networks, are often used by systems biologists to express gene interactions [12]. Each node corresponds to a gene and the incoming edges to a node x_i reflect the genes that influence its behavior. The function f_i specifies the net effect its constituents have on a gene being considered. The values of the variables correspond to the amount of gene products, in the Boolean case, zero or one.

Given a genetic network and an initial vector for the values of the variables, it is straightforward to construct a transition graph T , whose nodes denote a *state* of the network and whose edges represent the valid transitions from one state to its successor. A *state* is the binary vector representing the values of the variables at a given time, the initial state corresponding to time zero. The functions f_i 's are used to determine the unique successor to a given state. The transition graph T represents the dynamic behavior of a network. Biologists are particularly interested in the existence of stationary states. A state is *stationary* when its successor is itself (see for example [6, 12, 13]).

Transition graphs are traversed by the program P by simply making its output become the next input. Such a program will loop in stationary states or when a previously traversed state is revisited. By tracking previously traversed states, the program P can be easily modified to construct T . Assuming n variables, the total number of possible states is 2^n . Since n can be large (hundreds of genes), the worst case scenario for reconstructing P requires a set of 2^n I - O pairs. This work demonstrates that, by selectively choosing the I 's it is possible to reduce considerably the number of needed pairs. This is particularly significant when the maximum number of predecessors of the nodes in G (i.e., its highest in-degree) is much smaller than n .

The proposed approach is based on the computation of Boolean or discrete Jacobian matrices. Like their continuous counterparts a Jacobian matrix expresses how the function f_i changes as the x_j variable changes. Note that the incidence matrix for the graph G shares the same zeros as the corresponding Jacobian matrix. In the case of Boolean functions f_i the corresponding Jacobian matrix is also Boolean.

The main contribution of this paper is to determine an upper bound on the number of appropriately chosen I - O pairs as a function of the number of variables n and the maximum in-degree k_{\max} of the nodes in G . An additional contribution in this paper is to relate our method to others that have appeared in the literature and suggest possible ways of combining different approaches.

The methodology used in this work can be extended to the case of programs whose variables are integers within a known range. We will see that the functions f_i can be expressed by a vector yielding the functions' value when given input values for its components.

Sections 2 and 3 deal respectively with the assumed properties of the program P and the representation of the functions f_i . Section 4 describes related work. The material in Sections 5 and 6 addresses the definition of discrete Jacobians and how they are utilized to represent the incidence matrices of the graphs G . The formal description of our approach appears in Section 7. Section 8 presents an upper bound for the number of Jacobians needed to infer G (or P) from the I - O data. The proposed algorithm is summarized in Section 9. The actual results stemming from the theoretical work are reported in Section 10, as well as a detailed example. Section 11 covers the extension of this work to the case of bounded integers. Finally, in Section 12 we discuss the limitations of the approach and how they may be surmounted.

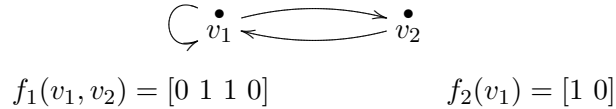
2. SYNCHRONICITY AND DYNAMIC BEHAVIOR PATTERNS

It is assumed in this work that the transition from one state to its successor takes place regardless of how many elements of a state vector are changed. This type of transition is called *synchronous* and it is deterministic in the sense that there is only one successor to a given state.

The term *asynchronous* is used to denote transitions in which *only one element* of the state vector is changed at a time. This implies that if there are multiple changes from a vector state to its successor, each of these changes takes place asynchronously at different times, the net result being that there are many possible successors to a given state; each of these successors is non-deterministically accessible from the original state. The assumption used by many computational biologists is the asynchronous one, based on the argument that, in a cell and in nature in general, simultaneous changes rarely occur. Nevertheless to our knowledge, the reverse engineering approaches that have been proposed in the literature (see Section 4) are based on synchronous transitions. This is also the case when we view reverse engineering as program synthesis from data.

It is important to distinguish two kinds of data. The first is simply an I - O pair, the second is a branch of the transition graph T , i.e., a cascading sequence of I - O pairs such that the O of a pair is the I of the successor pair. In this sense a set of I - O pairs may or may not contain cascading sequences.

In our approach we synthesize the program P from a set of I - O pairs. However, we will see in Section 12 that one may add requirements that constrain the generated program P to be such that its T contains a given cascading sequence of I - O pairs.

FIGURE 1. An example of the decorated graph G

3. REPRESENTATION OF BOOLEAN FUNCTIONS

Boolean formulas having a small number of variables can be effectively represented by the formulas' truth table. The advantage of this representation is that it avoids parenthesized notations using the Boolean operators for disjunction, conjunction and negation.

The truth table representation is particularly useful when attempting to generate a candidate Boolean formula that may correspond to partial available data. Consider for example the case of the formula (*not*(p) and q) or (p and *not*(q)), which is equivalent to the *exclusive-or* operator. Its truth table representation is the vector $[0 \ 1 \ 1 \ 0]$; it is implicitly assumed that the individual values of this vector correspond to the values of the pairs (p, q) in the order 00, 01, 10, 11. When k variables are used, the vector representation requires 2^k bits; we will refer to the vector representing a Boolean formula with k variables as the k -vector of the formula. Given a k -vector it is always possible to find the equivalent formula (or formulas) using methods that are well known in circuit design

With the current hardware advances in memory cost, bit manipulation and indexing capabilities, the truth table representation allows the fast computation of a Boolean formula in constant time.

Obviously the k -vector representation can be used in expressing the programs P that interest us. Equivalently, the graph G can be depicted by decorating each node with the corresponding k -vector and a list of the predecessors of a given node involved in the representation.

Figure 1 depicts one such graph with two nodes corresponding to two variables v_1 and v_2 . The Boolean formula decorating the node v_1 is (v_1 *exclusive-or* v_2) and that of node v_2 is simply *not*(v_1). The figure shows the k -vectors associated to each node.

The program P corresponding to the graph of Figure 1 is therefore:

$$\begin{aligned} v'_1 &\leftarrow (v_1 \text{ exclusive-or } v_2) \\ v'_2 &\leftarrow \text{not}(v_1) \end{aligned}$$

Our goal is to reconstruct P using a very limited set of I - O pairs.

4. RELATED WORK

In the past two decades the problem of the reverse engineering of genetic networks (GN) has been of great interest to systems biologists. This interest stems from two factors: (1) the pressing need to understand the dynamic

behavior of cells, and (2) the availability of micro-array data expressing how gene products evolve with time.

The micro-array data can be directly related to the transition graphs T mentioned in the introduction. The state vector associated with each node of T corresponds to the amounts of gene products at a given time. In the case of Boolean GNs these amounts are either zero or one. A sequence of transition states of T are the snapshots of gene products as time progresses and it is the information that can be extracted from micro-array data.

The reverse engineering problem consists of deducing G from data obtained by micro-array experiments. These experiments start at a given state of the cell and usually proceed until stationary states are reached; alternatively the experiments detect oscillations that correspond to the cycles that may exist in T . The amount of data is limited and often the induced graph G has to be revised as new experiments rule out or confirm the dynamic behavior of G . The limited amount of data consists of a set of pairs of I - O of the program P that we wish to synthesize. We will call a pGNT a table of an unknown GN having a partial number of arbitrary I - O pairs. A full GNT is the complete set of I - O pairs whose size is 2^n .

There are many articles in the literature on reverse engineering of genetic networks. A great deal of the approaches suggested in those papers requires exhaustive searches using trial-and-error to infer the desired graph G from an arbitrarily chosen pGNT. Our method differs from the existing ones in the sense that we call for a number of *specific* I - O pairs to be present in the pGNT. In Section 8 we will see that for graphs G having small in-degrees the inference can be done in polynomial time. In what follows we briefly review some of the proposed approaches that deal with exhaustive searches and contrast them with our method.

Akutsu et al. [2] proposed an interesting algorithm that assumes a given k_{\max} , i.e., the highest in-degree of all the nodes in G . This assumption implies that all the k -vectors are of the form $[K_1, K_2, \dots, K_{2^{k_{\max}}}]$ where the K 's are distinct Boolean variables that vary from node to node.

The algorithm tries to assign all possible values (one and zero) to each K in the k -vector of each node, by consulting the existing I - O pairs in the pGNT. If consistency is achieved, i.e., there are no conflicts in assigning the K 's, and if all the K 's are bound to either one or zero the algorithm terminates successfully. If no assignments succeed the algorithm fails.

Essentially Akutsu's algorithm tries to generate consistent Boolean formulas (k -vectors) for each node of G that satisfy the given contents of the pGNT. A further step in the algorithm is to collapse the k -vectors associated to each node by checking if a variable is fictitious. For example consider the 2-vector $[0 \ 1 \ 0 \ 1]$ representing the Boolean function $f(v_1, v_2)$. It can be collapsed into $f(v_2) = [0 \ 1]$ since the variable v_1 is fictitious. This amounts to having found that an assumed predecessor of a given node is actually irrelevant in affecting that node.

Their algorithm does not allow multiple k -vectors to be attached to a node of G . Such a situation arises from having k -vectors whose variables are not instantiated. One could easily modify Akutsu’s algorithm to allow uninstantiated variables. Such modification could be used to count the number of possible graphs G that satisfy a given pGNT.

Let m be the number of entries in the pGNT. Akutsu et al. proved that this algorithm utilizes $O(k \cdot 2^{2^k} \cdot n^{k+1} \cdot m)$ checks for consistency and that it requires a partial GNT of size $O(2^{2^k} \cdot (2k + \alpha) \cdot \log n)$ to find a unique solution with probability $1 - 1/n^\alpha$.

Simulated computer experiments in [2] show that for small values of k_{\max} (2 or 3) a surprisingly small size of the pGNT is often needed to achieve consistent solutions. For example, when $k_{\max} = 3$, about one hundred arbitrarily chosen I - O pairs are usually sufficient to obtain a consistent solution in (random) GNs having about 80 nodes.

The approach proposed by Liang, Fuhrman and Somogyi [8] can be viewed as being related to Akutsu’s. Instead of the consistency criteria advocated by Akutsu et al., Liang Fuhrman and Somogyi use information-theoretic measures to determine the predecessors of a given node that comply with a pGNT. As in [2] an exhaustive search is used for this purpose and the user has to specify the maximum in-degree to be used in the computations.

The concept of perturbation used by Ideker, Thorsson and Karp [7] requires that additional laboratory experiments be performed to determine the results of perturbing the output of a gene product. Like Liang, Fuhrman and Somogy [8] these authors also use information theoretic measures to determine the k -vectors. Their method is only applicable in inferring G ’s that have no directed loops, that is, they are restricted to DAGs.

Our method does not require that G be a DAG. On the other hand we require specific I - O pairs to be available. Once this is done the graph G is inferred deterministically, that is, without resorting to trial-and-error techniques.

In a sequel paper to [2], Akutsu, Kuhara, Maruyama and Miyano [1] consider an “experiment” to be an I - O pair in which a specified number of nodes in the input sequence have values determined by the experimenter and the remaining nodes have random values. The “cost” of such an experiment is the number of nodes specified by the experimenter to have a given value, 0 or 1.

The authors of [1] then determine upper and lower bounds on the number of experiments needed to infer a genetic network, assuming the cost is bounded by some constant C . It will be seen in the following sections that, in our approach, we allow the values of all nodes to be specified by the experimenter. It will be shown in section 10 that, under this last assumption, the number of I - O pairs necessary to infer the network is significantly reduced from those reported in [2] and [1]. From a biologist’s perspective our

	Input			Output			
	v_1	v_2	v_3	v_1	v_2	v_3	
I_1	0	0	0	1	1	0	O_1
I_2	0	1	1	1	0	0	O_2
I_3	1	0	1	1	0	1	O_3
I_4	1	1	0	1	1	0	O_4
I_5	1	1	1	1	0	0	O_5

TABLE 1. Partial GNT

$$J(1, 1, 1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \begin{array}{ccc} \bullet & & \bullet \\ v_1 & & v_2 \end{array} \begin{array}{c} \xrightarrow{\quad} \\ \xleftarrow{\quad} \end{array} \begin{array}{c} \bullet \\ v_3 \end{array}$$

FIGURE 2. Jacobian computed on state $(1, 1, 1)$ using pGNT from Table 1

approach is admittedly less realistic than that of attributing costs to experiments. However, this situation may change in the future as new laboratory techniques are developed.

5. DEFINING JACOBIANS FOR BOOLEAN FUNCTIONS

Shih and Dong [10] introduced the notion of discrete Jacobians to study a problem in automata networks. In this section and in the next we show how these Jacobians can be used to reduce significantly the number of I - O pairs needed to synthesize the class of programs that interest us.

A discrete Jacobian at state \mathbf{x} is an $n \times n$ matrix defined by

$$J(\mathbf{x})_{i,j} = \begin{cases} 1 & \text{if } f_i(\mathbf{x}) \neq f_i(\mathbf{x} + \delta x_j) \\ 0 & \text{otherwise} \end{cases}$$

where \mathbf{x} and $\mathbf{x} + \delta x_j$ are two vectors differing only in the j th operand. Therefore $J(x)_{i,j} = 1$ when node j has an effect on node i at state \mathbf{x} , that is, when $j \rightarrow i$ is an edge of the genetic regulatory graph. Thus, the union of the Jacobians $J(\mathbf{x})$ over all states \mathbf{x} is the transpose of the incidence matrix for the graph G .

The computation of a Jacobian corresponding to one given state requires that the pGNT include all surrounding states (states that differ on only one bit). For example using the pGNT from Table 1 we can only compute a Jacobian on state $(v_1, v_2, v_3) = (1, 1, 1)$. The result is shown in Figure 2. The accompanying graph shows only the edges arising from this Jacobian. Jacobians of other states could give rise to additional edges.

6. COMBINING INCIDENCE MATRICES OF JACOBIANS

Figure 2 of the previous section illustrates the result of interpreting a Jacobian of a given state as an incidence matrix. A continuous version of these Jacobians was used by C. Soulé [11] in 2005 to prove a conjecture of Thomas [12] relating the dynamic behavior of a genetic system to properties of the Jacobian at a single state.

The Boolean assignments introduced in the first section of the paper can be written as

$$x_i[t + 1] \leftarrow f_i(\mathbf{x}[t]).$$

These assignments relate the value of a variable \mathbf{x} at time $t + 1$ to that at time t . Intuitively, they correspond to the discrete versions of the systems of differential equations considered by Soulé. His Jacobian matrices contain positive, negative and zero values. Associated to the Jacobian at a state \mathbf{x} is a (signed) interaction graph. Soulé has shown that, in the continuous case, the existence of a positive circuit (i.e., a circuit in which the product of the associated signs is positive) in the interaction graph at a single state is a necessary condition for the system to have multiple stationary states. More recently Remy, Ruet and Thieffry [9] have proved an analogous theorem in the discrete case.

We note that even though the results of Soulé, Remy et al. are remarkable from a theoretical perspective they are not realistic from a practical standpoint since there is no way to determine the relevant state without first computing the Jacobians. In the discrete case, the computation of the Jacobians for every state is of exponential complexity.

Our proposed approach uses Jacobians to reconstruct the entire network graph, including the k -vectors, using a drastically reduced number of states whose Jacobians are needed to determine the incidence matrix of the graph G .

7. FORMAL DESCRIPTION

In this section we provide the definitions and formalism that allow us to reduce and determine an upper bound on the number of Jacobians needed to infer P from a set of specified I - O pairs.

Recall that a genetic network is a directed graph together with a k -vector specifying the function f_i at each vertex v_i . For clarity, we will denote the underlying graph by G and the graph decorated with k -vectors by G^{vector} .

Consider a genetic network with n nodes V and any number of edges E . Let us denote the predecessors of a node j in V as $G^-(j)$. Let k be the maximum in-degree of G . Then the subset $G^-(j)$ can be specified by a binary string of length n containing at most k ones.

For any set U , let \mathcal{F}_U represent all possible binary strings indicating all subsets of a given set U . We will call such a string an *expression pattern* for U . In particular, expression patterns for V represent a state in the dynamic behavior of the program to be inferred. If $\phi \in \mathcal{F}_V$ is an expression pattern

for V corresponding to a subset $S \subseteq V$, and U is any subset of V , then the expression pattern $\phi_U \in \mathcal{F}_U$ corresponding to $S \cap U$ is called the *restriction* of ϕ to U .

Given an n and a k , our approach produces the minimal set of states for which the Jacobians need to be determined. This corresponds to computing a specific set of I - O pairs that guarantees recovering the graph G and the Boolean formulas (provided as k -vectors) that are attached to each node. In contrast with the algorithm in Section 4, ours is deterministic and it is guaranteed to produce the desired graph, once the specified I - O pairs are determined.

Proposition 7.1. *Let Ψ be a set of expression patterns for V whose restrictions to every subset $U \subset V$ of size $k - 1$ includes all possible expression patterns for U . That is,*

$$(1) \quad \forall U \subset V, |U| = k - 1, \forall \phi \in \mathcal{F}_U, \text{ there exists } \psi \in \Psi \text{ with } \psi_U = \phi.$$

Then any genetic network G^{vector} of maximum in-degree k can be recovered from the I - O data for the states Ψ and their Jacobians. This condition is also necessary; that is, a set of states Ψ whose Jacobians uniquely determine every genetic network of maximum in-degree k , must satisfy condition (1).

Proof. Let G^{vector} be a genetic network of maximum in-degree k and consider an edge $v_j \rightarrow v_i$. Since the incidence matrix for G is the transpose of the sum of the Jacobians, there is some state α whose Jacobian detects this edge, that is, $J(\alpha)_{i,j} = f_i(\alpha) + f_i(\alpha + \delta x_j) = 1$. Since f_i depends only on $G^-(i)$, the same will be true for any state whose restriction to $G^-(i) \setminus \{v_j\}$ agrees with α . In particular, since $|G^-(i)| \leq k$, there exists $\psi \in \Psi$ which restricts to α on $G^-(i) \setminus \{v_j\}$, hence $J(\psi)_{i,j} = 1$. Thus, we can recover all the edges in the graph G from these Jacobians. To recover the k -vectors, note that as ψ comprises all elements of Ψ , the restrictions of ψ and $\psi + \delta x_j$ to $G^-(i)$ give all possible expression patterns for $G^-(i)$. Hence the output data at these states determine the k -vector at v_i .

To prove the last statement of the proposition, consider a set Ψ for which condition (1) fails, i.e., there exists a subset $U \subset V$, $|U| = k - 1$, and an expression pattern $\alpha \in \mathcal{F}_U$ that does not appear as the restriction of any element of Ψ . We will construct a genetic network G^{vector} which cannot be reconstructed from the Jacobians of Ψ .

Choose $v_i \notin U$. Construct a graph with $G^-(i) = U \cup \{v_i\}$ and set

$$\begin{aligned} f_1(\phi) &= f_1(\phi + \delta v_i) \text{ for all } \phi \text{ with } \phi_U \neq \alpha, \text{ and} \\ f_1(\phi) &\neq f_1(\phi + \delta v_i) \text{ for all } \phi \text{ with } \phi_U = \alpha. \end{aligned}$$

Then the edge $v_i \rightarrow v_1$ is essential, but it is not detected by the Jacobian of any state $\phi \in \Psi$. \square

Intuitively, it seems that in Proposition 7.1 one should need all expression patterns for subsets of size k (not $k - 1$) since k incoming nodes can effect the outcome at any given node. However, if we fix the value of any $k - 1$

Set 1			Set 2		
v_1	v_2	v_3	v_1	v_2	v_3
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	1	0	0
1	1	0	1	1	1
(a)			(b)		

FIGURE 3. For $n = k = 3$, condition (1) of Proposition 7.1 is satisfied by Set 1 but is not satisfied by Set 2 because assignments $v_2 = 0, v_3 = 1$ and $v_2 = 1, v_3 = 0$ do not appear in Set 2.

of these incoming nodes, then computing the Jacobian requires *both* values of the k -th node, thus the I - O pairs involved in the Jacobian computations for the set Ψ contain all the required information.

Proposition 7.1 assumes that the in-degree of the genetic network is known. This is important. If k is taken to be less than the true in-degree, the Jacobians of the states Ψ will detect only some of the incoming edges and may thus produce incorrect Boolean functions. There is no way to detect this error from the algorithm itself. For example, even if two consecutive values of k give rise to the same results, this is no guarantee that the correct in-degree has been reached.

Example Figure 3 illustrates two examples of sets Ψ satisfying or not satisfying Proposition 7.1. Table 2 displays the states Ψ for $n = 5$ and $k = 3$.

8. THE NUMBER OF REQUIRED JACOBIANS

We now proceed to determine an upper bound on the size of the set of states satisfying condition (1) of Proposition 7.1. Define a set of expression patterns Ψ for V as follows. If k is even, let Ψ be the set of all expression patterns containing exactly $(k - 2)/2$ zeroes or exactly $(k - 2)/2$ ones. If k is odd, let Ψ be the set of all expression patterns for V containing exactly $(k - 1)/2$ zeroes or exactly $(k - 3)/2$ ones.

Proposition 8.1. *The set of expression patterns Ψ satisfies condition (1) of Proposition 7.1 provided $|V| \geq (3k - 4)/2$ for k even, or $|V| \geq (3k - 3)/2$ for k odd.*

Proof. Suppose k is even and $|V| \geq (3k - 4)/2$. Then for any $U \subset V$ with $|U| = k - 1$, and any expression pattern $\phi \in \mathcal{F}_U$, either the number of zeroes or the number of ones appearing in ϕ is less than or equal to $(k - 2)/2$. Since $|V - U| \geq (k - 2)/2$, we can find an expression pattern ψ for V with *exactly* $(k - 2)/2$ zeroes or ones that restricts to ϕ on U .

v_1	v_2	v_3	v_4	v_5
0	1	1	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0
0	0	0	0	0

TABLE 2. Set of states Ψ for $n = 5$ and $k = 3$

The odd case is similar, except that in this case ϕ contains at most $(k - 1)/2$ zeroes or at most $(k - 3)/2$ ones. \square

In Table 2, we illustrate the set of states Ψ described in Proposition 8.1 for $n = 5$ and $k = 3$. Note that for any pair v_i, v_j , all expression patterns for the pair can be found in these states.

Corollary 8.2. *Let $\mathfrak{M}(n, k)$ be the cardinality of the smallest set Ψ which satisfies condition (1) of Proposition 7.1 for a network containing n nodes of maximum in-degree k . We have:*

$$\mathfrak{M}(n, k) \leq \begin{cases} 2^{\binom{n}{(k-2)/2}} & \text{when } k \text{ is even and } n \geq (3k - 3)/2 \\ \binom{n+1}{(k-1)/2} & \text{when } k \text{ is odd and } n \geq (3k - 4)/2 \end{cases}$$

Proof. An upper bound on $\mathfrak{M}(n, k)$ is given by the cardinality of the set Ψ in Proposition 8.1. For k even, this cardinality is easily seen to be $2^{\binom{n}{(k-2)/2}}$ and for k odd, the cardinality is $\binom{n}{(k-1)/2} + \binom{n}{(k-3)/2} = \binom{n+1}{(k-1)/2}$. \square

We remark that these formulas hold for $k = 1, 2$ using the standard convention that $\binom{n}{0} = 1$.

9. THE PROPOSED ALGORITHM

Let Ψ be the set of expression patterns from Proposition 8.1. It follows from Propositions 8.1 and 7.1 that the incidence matrix M for G can be computed by the following simple algorithm. Let r be number of states in Ψ and let ψ_i be the i th state.

Algorithm 1

- 1: Set initial matrix M to empty
 - 2: **for** $i = 1$ to r **do**
 - 3: **determine** $J(\psi_i)$
 - 4: $M \leftarrow M \cup J(\psi_i)$
-

The transpose of the final matrix M yields the desired results. The space complexity of the algorithm is $O(n^2)$. The time complexity may be reduced

by inserting a test just after updating M : the loop computation can be shortened if any of the rows of M contain k non-zero entries. If this occurs then the maximum in-degree for the corresponding node has been reached and the subsequent computations of the Jacobians can be performed omitting that column.

It is important to recall, however, that to determine the k -vectors, one has to consistently retain sufficient I - O data as the Jacobians are computed. (See example in Figure 5.) This of course raises the space complexity according to the number of required I - O pairs.

If we choose Ψ to be as in Proposition 8.1, an estimate of the number of I - O entries needed to compute the Jacobians is as follows. Each ψ_i contains n bits and a Jacobian is obtained by successively changing the j th bit from a zero to a one (or from a one to a zero) for each j . For k even and $p = (k - 2)/2$, Ψ consists of sequences that have exactly p zeroes or exactly p ones. All the Jacobians of these sequences can be computed from the I - O data for sequences with exactly $p - 1$, p , or $p + 1$ zeroes or ones, respectively. Counting these, we conclude that

- for k even, $p = (k - 2)/2$, the number of I - O pairs required to compute the necessary Jacobians is

$$2 \left[\binom{n}{p-1} + \binom{n}{p} + \binom{n}{p+1} \right] = 2 \left[\binom{n+1}{p} + \binom{n}{p+1} \right].$$

For k odd and $p = (k - 1)/2$, Ψ consists of sequences that have exactly p zeroes or exactly $p - 1$ ones. To compute Jacobians for these sequences, we need I - O data for sequences with $p - 1$, p , or $p + 1$ zeroes and sequences with $p - 2$, $p - 1$, or p ones. In this case, we have

- for k odd, $p = (k - 1)/2$, the number of I - O pairs required to compute the necessary Jacobians is

$$\begin{aligned} \binom{n}{p-2} + 2\binom{n}{p-1} + 2\binom{n}{p} + \binom{n}{p+1} &= \binom{n+1}{p-1} + \binom{n+1}{p} + \binom{n+1}{p+1} \\ &= \binom{n+2}{p} + \binom{n+1}{p+1}. \end{aligned}$$

For example, when $k = 2$, $p = 0$ and the required number of I - O pairs is $2(n + 1)$. When $k = 3$, $p = 1$ and the required number of I - O pairs is $(n^2 + 3n + 4)/2$.

10. RESULTS

Figure 4 shows, using a logarithmic scale, the surface $\mathfrak{M}(n, k)$. For example, the growth in n when $k = 2, 3, 4$, or 5 is respectively $\mathfrak{M}(n, 2) \leq 2$, $\mathfrak{M}(n, 3) \leq n + 1$, $\mathfrak{M}(n, 4) \leq 2n$ and $\mathfrak{M}(n, 5) \leq n(n + 1)/2$.

In the general case, for a graph containing n nodes and of maximum in-degree k the size of Ψ is $O(n^{\lfloor (k-1)/2 \rfloor})$. By the discussion in Section 9, the set of I - O pairs required to compute the Jacobians of Ψ has size $O(n^{\lfloor (k-1)/2 \rfloor + 1})$. Each Jacobian requires n^2 steps to compute so the complexity is $O(n^{\lfloor (k+1)/2 \rfloor})$. Any algorithm based on an exhaustive search of the possible subsets of predecessors like the ones in Section 4 has a complexity

k	Jacobian algorithm	exhaustive search
1	n^2	n^2
2	n^2	n^3
3	n^3	n^4
4	n^3	n^5
5	n^4	n^6
6	n^4	n^7

TABLE 3. Table comparing the complexity from our proposed algorithm using Jacobians and algorithms using an exhaustive search on the predecessors

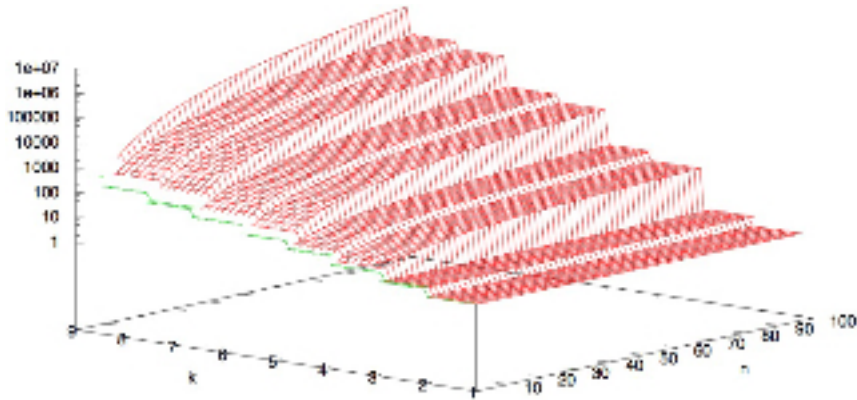


FIGURE 4. Minimum number of Jacobians needed to recover a graph with n nodes and of maximum in-degree k

of at least $O(n^{k+1})$ because it must test for every node in the graph all $\binom{n}{k}$ possible subsets of predecessors.

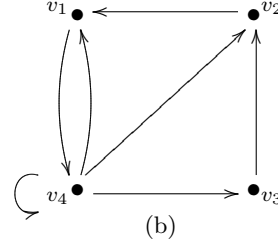
In Table 3 we compare the proposed algorithm with those that use exhaustive search. The ratio of the complexities between the exhaustive search algorithms and the one we proposed is $O(n^{\lfloor k/2 \rfloor})$. Nevertheless, note that this enticing result requires the pGNT to be specific instead of random.

Figure 4 shows that for the values $k = 2, 3$ and for n as large as 100 the number of needed Jacobians is about 2 and 101, respectively.

As it often happens when comparing different algorithms designed to perform similar tasks, there are trade-offs between time and space complexities between the exhaustive search algorithms and the one proposed in this paper.

The time complexity of our algorithm is lower than the existing ones using exhaustive search. However, as expected, our algorithm using Jacobians generally requires considerably higher space complexity than the one proposed in [2].

Input				Output			
v_1	v_2	v_3	v_4	v_1	v_2	v_3	v_4
0	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	0	1	0	1	1	0	0
0	0	0	1	1	1	1	0
0	1	1	1	0	1	1	0
1	0	1	1	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	0	1	1	0	0
1	1	1	1	0	1	1	1

(a) Specified set of I - O pairs

(b)

FIGURE 5. A network example for $k = 2$ and $n = 4$, with $f_1(v_2, v_4) = [1 \ 1 \ 1 \ 0]$, $f_2(v_3, v_4) = [0 \ 1 \ 1 \ 1]$, $f_3(v_4) = [0 \ 1]$ and $f_4(v_1, v_4) = [0 \ 0 \ 0 \ 1]$

$$J(0, 0, 0, 0) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

(a) Jacobian matrix computed in state $(0,0,0,0)$

$$J(1, 1, 1, 1) = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

(b) Jacobian matrix computed in state $(1,1,1,1)$

FIGURE 6. Jacobians computed using the specified set of I - O pairs from Figure 5

Notice also that our algorithm assumes the explicit knowledge of the value of k , the maximum assumed in-degree of the graph being inferred. So it is recommended that the user select a larger value of k to develop a theoretical model that is supported by known experiments. Obviously, the model will have to be revised with greater values of k , if a new experiment reveals the unsuitability of a current model.

Example Figure 5(a) provides an example of the specified set of I - O pairs with $n = 4$ and $k = 2$. The data in the table allows us to compute only 2 Jacobians, those for the two complementary states $(1, 1, 1, 1)$ and $(0, 0, 0, 0)$, and thus to infer the G^{vector} graph in Figure 5(b). The Jacobians for states $(1, 1, 1, 1)$ and $(0, 0, 0, 0)$ are shown on Figure 6. Note that the incidence matrix for the graph is the sum of the Jacobians.

11. EXTENSION TO A RELATED CLASS OF BOUNDED-INTEGER PROGRAMS

The proposed approach using Jacobians can be generalized to the synthesis of programs whose variables are bounded integers. For that purpose it is necessary to extend the notion of k -vectors and Jacobians to that domain.

When all the variables are integer and bounded in the interval $[0, q - 1]$ the corresponding k -vector is of size qk . An element of the vector can be indexed by a number in base q . The extension to the discrete Jacobians is done as suggested by Shih and Dong in [10].

The genetic networks using discrete threshold values as advocated by Thomas [12] and de Jong [6] can be inferred based on our approach. In this case, expression patterns $\psi \in \mathcal{F}_V^q$ are base q numbers of length n and to determine G^{vector} one requires Jacobians for a collection of expression patterns that restrict to all possible patterns on subsets of size $k - 1$.

12. FINAL REMARKS

We have shown that the described approach can reduce significantly the complexity of synthesizing the class of bounded-integer programs consisting of a sequence of assignments of functions of the variables to temporary variables \mathbf{x}' .

At the present time the approach using Jacobians is only applicable to small genetic networks. This is because it is currently difficult to induce a cell in a wet lab to re-start its dynamic behavior at a given state. This situation may well change in the future.

One possible way of circumventing the present difficulties in biological experiments is to develop programs to inspect the huge volume of microarray data to determine if there are instances of I - O pairs that can be used to construct pGNT's satisfying our requirements. We have explored the possibility of determining the probabilities of finding suitable Jacobians in random data but that would have little practical value since actual microarray data is far from being random.

The problem of inferring the graph G under the assumption of asynchronicity can also be explained in terms of programs. In this case the program P is considerably more complex than a program exhibiting synchronous behavior since the sequential computation of the \mathbf{x}' should be bypassed as soon as the computed value of the Boolean function reveals that the previous value of \mathbf{x} has been changed.

To simulate a program mimicking asynchronous behavior, P would have to be called recursively with two parameters, the vectors \mathbf{x}' and \mathbf{x} . A recursive call takes place when a given computed \mathbf{x}' differs from the corresponding \mathbf{x} . The new call considers new variables representing the \mathbf{x}' and the previously (and partially) computed \mathbf{x}' as the given \mathbf{x} . This arrangement corresponds to a non-deterministic program which is considerably more difficult to infer; this explains why the proposals for reverse engineering discrete genetic networks only deal with synchronous behavior.

In the remainder of this section we explore the combination of the methods briefly described in Section 4 with our approach using Jacobians. In that section we mentioned that Akutsu’s algorithm can be modified to produce multiple graphs G which are consistent with a given pGNT.

Assuming that not all the needed Jacobians are available it should be possible to generate, à la Akutsu, all graphs G that satisfy the pGNT. Once this is done current methods of model checking [4] can be used to test if any of the candidate graphs exhibits a desirable pattern of dynamic behavior that has been found in a laboratory experiment. A dynamic pattern corresponds to a cascading set of I - O ’s as described in Section 2. The use of dynamic patterns in the analysis of genetic networks has been advocated by several researchers including [3].

The above paragraph brings out the interesting duality that exists between examining a program at compile time (the graph G) and the programs dynamic behavior (the transition graph T). The research in abstract interpretation [5] addresses these problems and may well yield interesting results in reverse engineering problems like the one considered in this paper.

REFERENCES

- [1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano. Identification of genetic networks by strategic gene disruptions and overexpressions under a boolean model. *Theoretical Computer Science*, 2003.
- [2] T. Akutsu, S. Miyano, and S. Kuhara. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. *Pacific Symposium on Biocomputing*, pages 17–28, 1999.
- [3] G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: Extending thomas’ asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, 2004.
- [4] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
- [5] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. *Proceedings of the 4th Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, 1977.
- [6] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):69–105, 2002.
- [7] T. E. Ideker, V. Thorsson, and R. M. Karp. Discovery of regulatory interactions through perturbation: Inference and experimental design. *Pacific Symposium on Biocomputing*, 5:302–313, 2000.
- [8] S. Liang, S. Fuhrman, and R. Somogyi. Reveal: A general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing*, 3:18–29, 1998.
- [9] E. Remy, P. Ruet, and D. Thieffry. Graphic requirements for multistability and attractive cycles in a boolean dynamical framework. *Advances in Applied Mathematics*, 2005.
- [10] M.-H. Shih and J.-L. Dong. A combinatorial analogue of the jacobian problem in automata networks. *Advances in Applied Mathematics*, 34(1):30–46, 2005.
- [11] C. Soulé. Graphic requirements for multistationarity. *Complexus*, 1:123–133, 2003.

- [12] R. Thomas and R. D'Ari. *Biological feedback*. CRC Press, <http://hal.archives-ouvertes.fr/hal-00087681/fr/>, 1990.
- [13] R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. i. structural conditions of multistationarity and other nontrivial behavior. *Chaos*, pages 165–179, 2001.

R. CHARNEY, DEPARTMENT OF MATHEMATICS, BRANDEIS UNIVERSITY, WALTHAM,
MA 02453, USA

E-mail address: `charney@brandeis.edu`

J. COHEN, DEPARTMENT OF COMPUTER SCIENCE, BRANDEIS UNIVERSITY, WALTHAM,
MA 02453, USA

E-mail address: `jc@cs.brandeis.edu`

A. RIZK, INRIA PARIS-ROQUENCOURT, BP 105, 78153 L CHESNAY CEDEX, FRANCE

E-mail address: `Aurelien.Rizk@inria.fr`