2010 IEEE International Conference on Bioinformatics and Bioengineering

Finding Dynamic Modules of Biological Regulatory Networks

Ferhat Ay*†, Thang N. Dinh*, My T. Thai, Tamer Kahveci
Computer and Information Science and Engineering University of Florida Gainesville, FL 32611
*The first two authors, F. Ay and T. N. Dinh, contributed equally to this work.
†Corresponding author: fay@cise.uf.edu

***Abstract*—Often groups of genes in regulatory networks, also called modules, work collaboratively on similar functions. Mathematically, the modules in a regulatory network has often been thought as a group of genes that interact with each other significantly more than the rest of the network. Finding such modules is one of the fundamental problems in understanding gene regulation. In this paper, we develop a new approach to identify modules of genes with similar functions in biological regulatory networks (BRNs). Unlike existing methods, our method recognizes that there are different types of interactions (activation, inhibition), these interactions have directions and they take place only if the activity levels of the activating (or inhibiting) genes are above certain thresholds. Furthermore, it also considers that as a result of these interactions, the activity levels of the genes change over time even in the absence of external perturbations. Here we addresses both the dynamic behavior of gene activity levels and the different interaction types by an incremental algorithm that is scalable to the organism wide BRNs with many dynamic steps. Our experimental results suggest that our method can identify biologically meaningful modules that are missed by traditional approaches.**

***Keywords*-regulatory networks; dynamic modular structure;**

I. Introduction

The distribution of interactions between genes in regulatory and signaling networks (i.e., *Biological Regulatory Networks (BRNs)*), is not random. They form statistically significant connected subnetworks corresponding to various biological functions. Such groups of genes are also called *modules*. Recent studies have shown that biological networks exhibit modularity [1]–[6]. In these networks, the interactions and the entities of a module collectively describe how a certain biological function is performed for an organism.

Numerous methods have been proposed to identify modules for different types of biological networks such as metabolic networks [1], protein-protein interaction networks [2]–[4] and BRNs [5], [6]. These methods, however, have two important drawbacks when they are applied to BRNs. We first elaborate on these two drawbacks. We then discuss how we address these problems.

1) Ignoring interaction types and directions: Existing methods often consider a biological network as an undirected graph, where each molecule maps to a node and each interaction between a pair of molecules maps to an edge. In the resulting network all the edges are undirected and assumed to be of the same type. We call the networks constructed in this manner as *simplified networks* in the rest of this paper. This modeling strategy causes loss of biological context for BRNs where interactions have directions and different types (activation/inhibition).

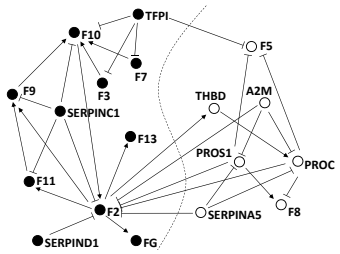

Figure 1. A portion of the human coagulation cascade network from KEGG Pathway Database [7]. Pointed arrow heads represent positive regulation (activation) and arrow heads with bars represent negative regulation (inhibition). The two divisions show the two modules found after simplifying the network as in existing methods.

Figure 1 shows an example of how this simplification degrades the accuracy of module identification. Consider the tissue factor pathway inhibitor (*TFPI*). It inhibits four different coagulation factors, namely *F3*, *F5*, *F7* and *F10*. Therefore, it acts as an anticoagulant when it is active. Indeed, it is annotated as a lipoprotein-associated coagulation inhibitor [7]. However, when these inhibitions are reduced to undirected edges as in Diao *et al.* [6], *TFPI* is grouped with most of the coagulation factors which act as coagulation activators. In other words, this simplification deteriorates the identification of correct functional decomposition.

2) Ignoring activity level changes of genes over time: The second weakness of existing methods is they ignore that the activity levels of the genes of a BRN can change over time due to interactions between genes. Existing methods often work for only static snapshots of BRNs. Consider the example in Figure 1. Here, if *THBD* is active at the current state, it will activate *PROC* in the next state which in turn will inhibit *F2*, *F5* and *F8*. These state changes create different snapshots of the BRN. The traditional approach to deal with such dynamic networks is to identify the modules from scratch for each snapshot [8]–[12]. However, coping with network snapshots independently may lead to

978-0-7695-4083-2/10 $26.00 © 2010 IEEE
DOI 10.1109/BIBE.2010.31

substantial variation in obtained modules in consecutive states, resulting in inconsistent modular structures. Several recent methods point out the importance of tracking the evolution of the modular structure through dynamic steps in other contexts [13]. Furthermore, incrementally updating modular structure is computationally more efficient than dealing with each snapshot separately.

In this paper, we design an algorithm that considers both the interaction types and the directions. It also allows incremental update of modules when the underlying BRN changes its state in time. This incremental strategy allows us to keep track of the evolution of individual modules and improves the running time. Below is the formal statement of the problem that we address in this work:

Problem definition: Given a BRN and an initial network state S^0 that consists of states of each gene (active or inactive), identify the sequence of module structures $\mathcal{C}^0$, $\mathcal{C}^1$, $\cdots$, $\mathcal{C}^t$ dynamically when the state of the network changes as S^0, $S^1, \cdots, S^t$ over time where $\mathcal{C}^i$ is the partitioning of the genes of input BRN into modules according to state S^i.

Our contributions: We develop a novel method to find the dynamic modules of a BRN when the state of the BRN can change over time. Our approach differs from the existing ones from the very beginning (i.e., the modeling phase). Instead of finding modules of *the simplified network* (i.e., ignoring edge types and directions), we first create a new network, named *functional network* from the underlying BRN for the each different state. The nodes of a functional network are the genes in the original BRN. The edges between two nodes represent the functional similarity of these genes at the corresponding state of the BRN. Starting from an initial state S^0 we use a state transition function to compute the next states of the BRN and at each state we update the functional network by considering the state changes of the genes. We observe that the functional networks at two consecutive states often show high similarity. Following this observation, we develop an incremental algorithm that computes the modules of the BRN at a new state using its modules in the previous state and the changes in the functional network. To further reduce the computational cost, we develop a compact representation of the network in which the module information is embedded and the modular structure is preserved.

In summary, our technical contributions are:

- We introduce the concept of *functional network* that represents the functional similarities between genes at a given state of a BRN.
- We propose an algorithm that identifies the modular structure of dynamic networks such as BRNs.
- We build a compact representation of the modular structure of the BRNs. This allows our method to scale for large network sizes with many dynamic steps.

The organization of the rest of this paper is as follows: Section II describes how we address the issues of existing methods and detailed theoretical analysis of our algorithm. The experimental results are illustrated in Section III. Section IV briefly concludes the paper.

II. METHODS

This section discusses the algorithm we develop for identifying dynamic modules of BRNs. Briefly, this section is organized as follows. Section II-A discusses the simulation of the dynamic behavior of BRNs by using a state transition model. Section II-B describes how we construct functional networks from original BRNs for a given state of the network. Section II-C presents our algorithm that incrementally updates the modular structure of a BRN using functional networks at different time steps.

A. State Transitions

The dynamic behavior of BRNs stems from the alterations in gene activity levels. These alterations are determined by the states of interacting genes. We denote the state of the ith gene at time t as $X_i(t)$ where $X_i(t) = 1$ means that the ith gene is "active" (high activity level) and $X_i(t) = 0$ means that it is "inactive" (low activity level). We denote the state of a BRN with n genes at time t using a vector with n entries as x^t= [$X_1(t)$, $\cdots$, $X_i(t)$, $\cdots$, $X_n(t)$].

The state of the genes can change over time due to internal regulations. Let A_i and I_i be the set of activators and inhibitors of the ith gene respectively. An activator (or inhibitor) is *active* when its state is 1. The following equation computes the next state of the ith gene from the activity values of the genes at state x^t.

$$X_i(t{+}1) = \begin{cases} 0 & \text{if } [\sum_{j\in A_i} X_j(t) - \sum_{j\in I_i} X_j(t)] < 0 \\ 1 & \text{if } [\sum_{j\in A_i} X_j(t) - \sum_{j\in I_i} X_j(t)] > 0 \\ X_i(t) & \text{if } [\sum_{j\in A_i} X_j(t) - \sum_{j\in I_i} X_j(t)] = 0 \end{cases}$$

B. Construction of Functional Networks

Recall that we define the modules as the set of genes that collectively serve for a biological function. To find such modules, the first question to be addressed is: How can we model the functional similarity between two genes? Here we build a biologically and statistically sound approach. We say that two genes have similar functions at a given state of the BRN if their impacts on the state of that BRN are similar.

For a given state of a BRN with n genes, we construct an undirected and weighted graph. We call this graph the *functional network*. Each node of this network corresponds to a gene in the BRN. The weight of an edge shows the functional similarity of the two genes connected by that edge. We build this functional network as follows. We first calculate the impact of each gene on the given state. We represent the impact of each gene by an $n \times 1$ vector. Then, for each gene pair, we calculate the similarity of their impact vectors. We elaborate on each of these steps next.

Calculation of Impact Vectors: We denote the impact of the ith gene on the network at time t with an $n \times 1$ vector

$Imp_i(t)$ named the *impact vector*. We compute this vector as follows. Let x^t= [$X_1(t)$, $\cdots$, $X_n(t)$] denote the state of a given BRN at time t. Also, let y^t= [$X_1(t)$, $\cdots$, $X_{i-1}(t)$, $1-X_i(t)$, $\cdots$, $X_n(t)$] denote the state obtained by flipping only the state of the ith gene. We compute the next states of both x^t and y^t by applying our state transition rule and represent them by x^{t+1} and y^{t+1} respectively. The state of the ith gene at time t can be equal to 0 or 1. Then we compute the impact vector of the ith gene as $Imp_i(t) = (-1)^{X_i(t)}(y^{t+1} - x^{t+1})$.

Intuitively, we are computing the difference between two state vectors at $t+1$ when ith gene is active and when it is inhibited at time t. The jth entry of $Imp_i(t)$ is 1 if both gene i and j had the same form of state changes (i.e., $0 \rightarrow 1$ or $1 \rightarrow 0$). This entry is 0 when flipping the state of ith gene does not effect the state of jth gene. If i and j had reverse state changes the jth entry of $Imp_i(t)$ is -1. Thus, the impact vector shows the set of genes that change activity levels by altering only the ith gene and how their activity levels change. For instance, let x^{t+1} = [0, 0, 1, 1] and y^{t+1} = [1, 1, 0, 1] for the first gene of a hypothetical network with four genes, then $Imp_1(t) = (-1)^0$ [1, 1, -1, 0]. This shows that the second gene is activated and the third gene is inhibited by the activation of the first gene. The fourth entry of $Imp_1(t)$ is zero meaning that the first gene has no state changing effect on the fourth gene at time t. Biologically, non-zero entries of $Imp_i(t)$ shows the genes whose states are sensitive to the activity level of the ith gene at time t.

Calculation of Impact Similarities: Having constructed the impact vectors for each gene in the network for a specific state, now we describe how we use these vectors to calculate the similarities between the functions of genes (i.e., the edge weights in functional network). For this purpose, we calculate the statistical significance of the similarity of their impact vectors as follows. Consider genes i and j at time t. Let a and b denote the numbers of nonzero entries of $n \times 1$ impact vectors $Imp_i(t)$ and $Imp_j(t)$, respectively. Let c denote the number of nonzero entries at the same positions of these two vectors with the same value. If both impact vectors have an equal value at position k, it means that both gene i and gene j can alter the state of the gene k in the same direction at the current state. We compute the impact similarity as the minus log probability of the number of such commonly affected genes. Formally, let K be the random variable that denotes the number of common nonzero entries in $Imp_i(t)$ and $Imp_j(t)$ assuming the nonzero entries are uniformly distributed in these vectors. Without loss of generality, we assume $a \geq b$. Then, we calculate the impact similarity of genes i and j at time t as $\text{Sim}(i,j) = -\log[\Pr(K \geq c)]$ where $\Pr(K \geq c)$ denotes the probability that the number of common nonzero entries is greater than or equal to observed value (i.e., c) and calculated as $\Pr(K \geq c) = \sum_{x=c}^{b} \frac{\binom{n-a}{b-x}\binom{a}{x}}{\binom{n}{b}}$.

Formal Definition of Functional Network: We conclude this section by formally defining the functional network. Given a BRN and its initial state S^0, the functional network of that BRN at state S^t is the undirected weighted graph $G^t = (V, E^t)$, where each gene of the BRN corresponds to a node in V and E^t is the set of edges in G^t. We compute the state S^t from S^0 by using the given state transition function. Then, we calculate the edge weight between the ith and jth nodes (i.e., Sim(i, j)) as their impact similarity at state S^t.

C. Identification of Dynamic Modules

As the state of a BRN changes from one state to the next, its functional network often changes slightly. In this section, we develop an algorithm that exploits this observation. It computes the modular structure of the BRN at its new state from the modules of its old state instead of recomputing it from scratch. To develop our incremental algorithm, we first build a compact representation of the modular structure.

Compact Representation of a Network:

Let G = (V, E) be the functional network of a BRN at a certain state together with the impact similarity function $w : E \rightarrow \mathbb{R}$ as explained in Section II-B. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ be the modular structure of G, where C_i represents the i^{th} module in G, $\bigcup_{i=1}^{k} C_i = V$, and $C_i \cap C_j = \emptyset, \forall i \neq j$. We construct a new network with an equivalent modular structure to G but contains significantly smaller number of nodes and edges. We use this network in place of G to reduce the running time of module identification as G changes over time.

We build the compact representation $G' = (V', E')$ of $G = (V, E)$ as follows. Let us define the function Φ over a module C_i as $\Phi(C_i) = \sum_{u,v \in C_i} w(u,v)$. For each module C_i in G, if C_i contains only one node, we create one node x_i in G'. If C_i contains more than one node, we create two nodes x_i, y_i in G' and an edge between them with an edge weight equal to $\frac{1}{2}\Phi(C_i)$. For each pair of modules (C_i, C_j) in G, we insert four edges (one for each pair of nodes in opposing modules) $(x_i, x_j), (x_i, y_j), (y_i, x_j), (y_i, y_j)$, each has an edge weight equal to one fourth of the sum of the edge weights between C_i and C_j. Figure 2 illustrates this construction on a simple example.

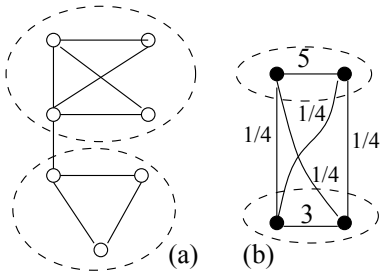

Figure 2. A hypothetical network with two modules (a) before compacting (b) after compacting. The module boundaries are shown dashed ovals. For simplicity, the weight of all the edges in (a) are one.

Here we show that this construction embeds the module information of G and preserves the modular structure. First let us recall the well-known modularity score ($\mathcal{Q}$) of Newman *et al.* [14]. Given a modular decomposition $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ $\mathcal{Q}(\mathcal{C}) = \sum_i \left[\frac{\Phi(C_i)}{2m} - \left(\frac{\text{vol}(C_i)}{2m}\right)^2\right]$ where m is the total weight of all edges in the network and $\text{vol}(C_i)$ is total weight of all edges that are incident to at least one node in C_i. We prove that the modular structure is preserved in compact representation in Theorem 1 by using the lemmas below. We omit the proofs of lemmas due to space limitation.

Lemma 1: (CONSERVATION OF MODULARITY SCORE) Let $G = (V, E)$ a network and $\mathcal{C}$ be its modular structure. Then, the compact representation $G' = (V', E')$ has the modular structure $\mathcal{C}'$ such that $\mathcal{Q}(\mathcal{C}') = \mathcal{Q}(\mathcal{C})$.□

Lemma 2: (CONSERVATION OF MODULE MEMBERSHIP) Let $G' = (V', E')$ be the compact representation of the network G. Also, let x_i and y_i, $x_i, y_i \in V'$, be the two nodes constructed to represent the module C_i of G while compressing G to G'. Then, for any modular decomposition $\mathcal{C}''$ of G' in which x_i, y_i belong to two different modules there exists a modular structure of G' with modularity score greater than $\mathcal{Q}(\mathcal{C}'')$ in which x_i, y_i belong to the same module. □

Theorem 1: (CONSERVATION OF MODULAR STRUCTURE) Given a functional network $G = (V, E)$ and its optimal modular structure $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ (i.e., $\mathcal{Q}(\mathcal{C})$ is maximum), let $G' = (V', E')$ be the compact representation of G and $\mathcal{C}' = \{C'_1, C'_2, \ldots, C'_k\}$ be its modular structure computed as described in this section. Then, $\mathcal{Q}(\mathcal{C}')$ is maximum over all possible modular decompositions and there is a one-to-one correspondence between $\mathcal{C}$ and $\mathcal{C}'$.

Proof: Lemma 1 states that $\mathcal{Q}(\mathcal{C}) = \mathcal{Q}(\mathcal{C}')$. Here we first show that there exists no other modular decomposition $\mathcal{C}'' = \{C''_1, C''_2, \ldots, C''_t\}$ of G' such that $\mathcal{C}'' \neq \mathcal{C}'$ and $\mathcal{Q}(\mathcal{C}'') > \mathcal{Q}(\mathcal{C}')$. Since $\mathcal{C}'' \neq \mathcal{C}'$, there exists at least two nodes x_i, y_i such that $x_i, y_i \in C'_i$ but $x_i \in C''_a$ and $y_i \in C''_b$ where $C''_a \neq C''_b$. By Lemma 2 we know that there exists another modular decomposition in which x_i, y_i belong to same module and the modularity score is greater than $\mathcal{Q}(\mathcal{C}'')$. Applying this argument iteratively we get $\mathcal{Q}(\mathcal{C}')$ is maximum over all possible modular decompositions. Also, again by Lemma 2 each $C'_i = \{x_i, y_i\}$ and corresponds to the original module C_i of G. Hence, there exists a one-to-one correspondence between $\mathcal{C}$ and $\mathcal{C}'$. ■

Incremental Identification of Dynamic Modular Structure:

Here we describe our algorithm that identifies the modules of BRNs at each state incrementally (i.e., without recomputing them from scratch) by utilizing the compact representation we devised. Formally, let S^0 be the initial state of a given BRN and $G^t = (V, E^t)$ denote the functional network of this BRN at state S^t where $w_t : E^t \rightarrow \mathbb{R}$ is a non-negative weight function. Also, let $\mathcal{C}^t = \{C_1^t, C_2^t, \ldots, C_k^t\}$ be the modular structure of G^t, $\forall t \geq 0$. The Algorithm 1 computes $\mathcal{C}^{t+1}$ using $\mathcal{C}^t$ and $G^t \oplus G^{t+1}$ where $\oplus$ is the symmetric difference symbol. At a high level, our method identifies the modules of dynamic BRNs by employing an external algorithm ($\mathcal{A}$) to calculate the modular structure of compact representation at each state. To do this, we use the well-known CNM algorithm [15] as $\mathcal{A}$.

We give a description of our incremental method in Algorithm 1. First, we compute the initial modular structure $\mathcal{C}^0$ and its compact representation $\mathcal{C}_c^0$ from the functional network G^0 at state S^0. After that, till the network reaches a steady state [16] or it visits a user defined number of states, for each state S^t we apply the following three main steps:
(1) Combine changes in comparison to the previous state and previous modular structure to update the compact representation of G^t (Algorithm 1, lines 5-16);
(2) Apply algorithm $\mathcal{A}$ on the compact representation to obtain its modular structure;
(3) Refine the obtained modular structure on the compact representation and decompress it to get the actual modular structure of the network.

In step (1), we update the compact representation as follows. Consider the set of nodes Δ_V that are incident to updated edges. The nodes of the set Δ_V are subject to changing their memberships. To allow these nodes to leave their previous modules and join new ones as the network state changes, we move all nodes in Δ_V out of their modules and treat each node as a singleton module. In other words, for each node $u \in \Delta_V$, we create a new node x_u as a new singleton module in the compact representation. For each new node x_u, if u is adjacent to any module C_i, we create two new edges (x_u, x_i) and (x_u, y_i) to connect this singleton module to C_i accordingly and assign the edge weights as half of the sum of edge weights between these two communities. For any C_i containing u, we also adjust the weight of edge (x_i, y_i) by subtracting the sum of edge weights from u to other nodes in C_i to reflect the removal of u from C_i.

After incorporating the changes into the compact representation, in step (2), we run algorithm $\mathcal{A}$ again on the new compact representation to obtain its modules. Note that the modular structure obtained at this step is in compact form. This allows us to reduce the running time of this step compared to uncompressed network.

In step (3), we further refine this modular structure based on Lemma 2 as follows. If x_i and y_i are assigned to different modules, we either move x_i to the module containing y_i or move y_i to the module containing x_i depending on which one increases the modularity more. This refinement makes sure that x_i, y_i are always assigned to the same module and doing so will increase the modularity value as we stated in Lemma 2. Finally, we decompress the compact modules to obtain the actual modules of the BRN at current state.

Algorithm 1 Incremental Algorithm for Identifying Dynamic Modules of a BRN

Compute the functional network G^0 using S^0
Find the modular structure $\mathcal{C}^0$ of G^0 using $\mathcal{A}$
Compute the compact modular structure C_c^0 of G^0
for each state $S^{t-1}, t > 0$ **do**
  Initialize C_c^t of G^t to C_c^{t-1}
  Use state transition function to compute S^t
  Let $\Delta_E = \{(u,v) | (u,v) \in \left(E^{t-1} \oplus E^t\right)\}$
  for all $e \in \Delta_E$ **do**
    Update edge weights between modules of C_c^t
  end for
  Let $\Delta_V = \{u | \exists v, (u,v) \in \left(E^{t-1} \oplus E^t\right)\}$
  for all $v \in \Delta_V$ **do**
    Extract v from its module and remove its edges
    Create a new singleton module that contains v
    Calculate the edge weights between modules of C_c^t and the new module
  end for
  Find the modular structure $\mathcal{C}_c^t = \{C_{c,1}^t, C_{c,2}^t, \ldots, C_{c,m}^t\}$ of G_c^t using Algorithm $\mathcal{A}$
  Refine the modular structure as stated in Lemma 2
  Decompress $\mathcal{C}_c^t$ to get $\mathcal{C}^t$
end for

Our incremental method described in this section has important advantages over the traditional methods which assume that BRNs are static networks and calculate modular structure depending on this assumption such as [2] and [6]. Firstly, we introduce the concept of functional network that takes into account both the different interaction types and their directions. Functional networks also allow us to simulate the dynamic behavior of the BRN through state transitions. Secondly, the incremental calculation of modular structures at different states makes it possible to track the evolution of modules (i.e., membership changes, creation of new modules). Tracking the module evolution is difficult when computing modular structure independently at each state. Also, the compact representation we use improves the running time of our algorithm by only considering the symmetric difference of consecutive states. For very large networks or networks with many dynamic steps compact representation scales the problem such that it can be handled efficiently by modularity identification algorithms.

III. RESULTS

In this section, we evaluate the accuracy and the performance of our method on real BRNs. We first compare the biological relevance of our results with the ones of existing methods that use simplified networks (i.e., the edge directions and types are ignored) [2], [6]. We use CNM method [15] both as the module identification method for these simplified networks and as the external algorithm that we employ to find the modular structure of compact versions of functional networks that we generate. The goal in this experimental setup is to see the merits of building functional networks without doing any simplification as traditional methods.

In the second set of experiments (Section III-B) we compare the effect of compressing the networks on the modularity score and the running time when network has a number of dynamic steps. For this purpose we apply CNM method on the network of each dynamic step individually and we compare it with the results we gather when we use CNM as our external algorithm to find the modular structures incrementally. It is important to note that the aim of this experiment is not to compare whether our algorithm or CNM is better. Instead, we analyze the effect of network compression in dynamic module identification.

Datasets: We use all the regulatory and signaling networks available in the KEGG pathway database for *H. Sapiens* [7]. There are totally 2103 genes and 9188 interactions in KEGG for *H. Sapiens*. We use gene expression data of breast cancer patients from the literature and compile four different datasets containing totally 722 patients [17]–[20]. We use these gene expression values to define the initial state of the network for each patient.

Environment: We run all the experiments on a desktop computer with one Intel Pentium 4, 3.20 GHz processor and 2 GB of RAM.

A. Qualitative evaluation

Here we evaluate the significance of the functional networks that we devised. We first discuss the results on a specific real example in detail. We then present the most frequently observed modules by our method for human regulatory network by using the gene expression data of 722 breast cancer patients.

Coagulation cascade network: A case study. Recall that in Figure 1, we have shown that the existing methods may fail to identify the biologically significant modules. Here, we revisit the same network (human coagulation cascade) and evaluate whether our functional network can overcome this drawback on a real example. Figure 3 illustrates two modular structures that our method identifies for two consecutive time steps of human coagulation cascade. Figure 3(a) shows that our method perfectly separates the genes into two modules that serve for coagulation and anti-coagulation functions. The anti-coagulation module has eight members. Alpha-2-macroglobulin (*A2M*) is a protease inhibitor and it inhibits thrombin which results in an adverse effect on clotting. *PROC* encodes protein C, a vitamin K-dependent plasma glycoprotein that is a key component of the anticoagulant system. *PROS1* has an anticoagulant effect too, as it is a cofactor of activated protein C. *THBD* activates *PROC* by binding to thrombin which results in degradation of the activated forms of coagulation factors *F5* and *F8*. *TFPI*

gene encodes a protease inhibitor that inhibits the activated coagulation factors of *F10* and *F7* in an autoregulatory loop. The other three genes of the anti-coagulation module are the three members of *SERPIN* family and they act as an inhibitor of thrombin through different mechanisms in different conditions.

The coagulation module consists of coagulation factors with names starting with *F*. The coagulation starts by *F7* coming in contact with tissue-factor and forming an active complex that activates *F9* and *F10*. *F10* and its cofactor *F5* forms a complex that activates prothrombin to thrombin which in turn activates other components of coagulation cascade (*F8*, *F9*, *F11*, etc.). Figure 3(a) shows that these coagulation factors are grouped together as a module and hence we call this the coagulation module. *Thus, we observe that by utilizing functional networks our algorithm can separate the genes of a BRN into biologically meaningful modules.*

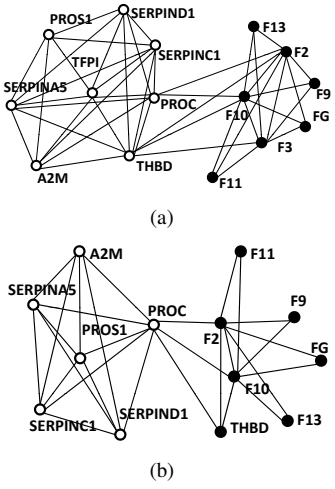

Figure 3. Two functional networks induced from human coagulation cascade at two consecutive time steps.

Our algorithm can update the modules incrementally as the state of the BRN changes. Figure 3 shows how the functional networks and the modules of the human coagulation cascade network evolve. The modules of the functional network change as the states of the genes change. In the first state (Figure 3(a)), the activity level of *F10* is low. In this case, the tissue factor pathway inhibitor (*TFPI*) can suppress the activating effect of *F3* on *F10*. Keeping the activity level of a coagulation factor low, *TFPI* has functional similarity to the other anticoagulants such as *PROC*, *PROS1* and *SERPINs* in this snapshot. However, in the next state (Figure 3(b)) *F10* becomes highly active and *TFPI*'s inhibiting effect on *F10* decreases. In this case, *TFPI* can not show anticoagulant effect, hence, it is no longer a member of anti-coagulation module. This change results in restructuring of the modules and our algorithm is able to identify the new modular structure reflecting the changes.

Evaluation on the entire human regulatory network: We apply our method to human regulatory network extracted from KEGG [7] and we use published gene expression data of 722 different breast cancer patients [17]–[20] to determine the initial states of the genes for each patient.

Table I
TOP 20 MODULES FOUND BY OUR METHOD WITH THE HIGHEST SUPPORT FROM THE 722 PATIENTS. MODULES IN **BOLD** ARE MISSED ENTIRELY BY EXISTING METHODS THAT USE SIMPLIFIED NETWORKS.

Rank	(%) Support	Genes of the module	(%) Precision (simplified networks)
1	100.0	ELK4, FOS, SRF	8.6
2	97.1	SMAD2, SMAD3, SMAD4 SMAD1, SMAD5, SMAD9	75.0
3	93.8	NGF, NTF3, NTF4 NTRK1, NTRK2, BDNF	100.0
4	92.7	IL24, IL20, IL22RA1	60.0
5	92.7	MAP2K3, MAP2K6	2.9
6	92.4	FRAP1, RPS6KB1 RPS6KB2, RPS6	50.0
7	91.0	CHUK, IKBKB, IKBKG NFKBIA, NFKBIB, NFKBIE	5.7
8	90.6	CAMK4, CREBBP, EP300	5.7
9	88.6	NFKB1, RELA, BCL2	2.9
10	87.0	ADCY3, GNAL, PRKG1	2.5
11	86.8	TNF, TNFRSF1A, TRADD	2.9
12	86.7	PRKAA1, PRKAA2, PRKAG2 PRKAG3, PRKAB1, PRKAB2 PRKAG1, SLC2A4	42.1
13	86.1	PARD3, IGSF5, F11R JAM2, JAM3	4.2
14	85.9	**JAK2, STAT3, POMC**	**0.0**
15	85.7	**WASF2, BAIAP2, ARPC5 ARPC1B, ARPC2, ARPC5L**	**0.0**
16	85.2	FIGF, VEGFC, FLT4	3.4
17	85.0	CHEK2, ATM	7.5
18	83.9	**RHOA, DIAPH1, PFN3 PFN4, PFN1, PFN2**	**0.0**
19	83.4	ADCY3, GNAL, PRKG2	2.5
20	83.1	RASGRP1, SOS1, SOS2	3.4

While traditional methods assume that each patient has the same snapshot of the regulatory network, we consider gene expression data to construct patient specific functional networks. Considering the variations in gene expression levels of different patients, our method allows us to identify the most frequent modules that are observed in a set of breast cancer patients.

We define the *support* of a module as the percentage of patients whose functional network created from initial gene expression values contains that module. In Table I, we list the top 20 modules with largest support. For the existing methods that use simplified networks, the topology of entire network is independent of the expression levels of the genes. As a result, the modules are exactly same for all patients and there is only one simplified network. Here we measure the precision of these methods as follows. For each significant module X from our method, we first find the module Y from the simplified network that contains X if there is any. Let $|X|$ and $|Y|$ be the number of genes in X and Y. We measure the precision for X as $100\times|X|/|Y|\%$. Thus, 100% means that CNM could identify the same module and 50% means that it identifies the module X inside a module that is twice the size of X. 0% implies that CNM does not group the genes of X in the same module.

Next, we discuss the biological relevance of several of these modules. The first module in Table I contains three genes from the MAPK signaling pathway. *FOS* takes part

in several other networks as well whereas the other two genes play role only in MAPK pathway according to the KEGG database. In this pathway *ELK4* can form a ternary nucleoprotein complex with the serum response factor (*SRF*) and *SRF* accessory protein 1 (*SAP1*) to activate *FOS*. As a result these genes collectively serve for cellular proliferation/differentiation. Proliferation is a well-conserved biological function among multicellular organisms. The genes listed in the second module in Table I are all from the *SMAD* family. They jointly appear in the TGF-beta signaling pathway though several of them take part in other pathways as well. On this pathway they collectively serve a critical role in a number of activities including cell growth, apoptosis, morphogenesis, development and immune responses [21]. The genes in the third module appear in the Neurotrophin signaling pathway. Studies have shown that the expression levels of these genes correlate significantly for patients that have damaged hippocampus as well as for healthy patients [22]. Our algorithm identifies *JAK3*, *STAT3* and *POMC* together as another module with high support. The genes in this module affect hypothalamo-pituitary-adrenal axis among a number of other functions. *JAK3* increases the activity of *STAT3* through phosphorylation. *STAT3* then indirectly activates *POMC*. However, simplified network approach fails to group these three genes together suggesting that their functional similarity is not significant.

The results of our qualitative evaluation has two important implications:
(i) Functional networks are useful in decomposing BRNs into modules that contain functionally similar genes.
(ii) Using gene expression data to create patient-specific networks allows identification of significant modules that are missed when simplified network approach is used.

B. Quantitative evaluation

In this section, we evaluate the performance of our method quantitatively. We want to see the effect of using compact representation on the quality of the modular structure and the running time of the method. We start with experimenting on whether our algorithm sacrifices modularity value $\mathcal{Q}$ as it uses a compact representation of the modular structure. Figure 4 plots the average modularity for each patient over all dynamic steps. The modularity of our method is close to that of CNM with negligible difference. The difference is below 3%. Hence, the loss of modularity score due to compression is not significant.

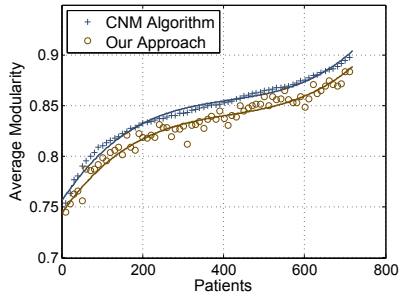

Figure 4. Average modularity value for each patient over all dynamic steps.

We further use the normalized mutual information (NMI) [23], an information-theoretical approach, to measure the similarity between the modular structures found by directly applying CNM algorithm on functional networks and first compressing these networks and then applying CNM. NMI takes a value in the range $[0, 1]$. A large NMI value implies that the two modular structures are similar. The results show that the NMI value is very close to 1 (mean NMI is 0.95), confirming the high similarity of two modular structures. It is important to note that the significant difference between modular structures in two different columns of Table I is due to the comparison of two different network types (functional network vs simplified network). Here the inputs are the functional networks and their compact forms. As Figure 5 suggests, the membership of the genes in modules in this case are highly similar. Therefore, the figures 4 and 5 together imply that for very large networks with many dynamic steps, where running time is an issue, using compact representation is a practical option.

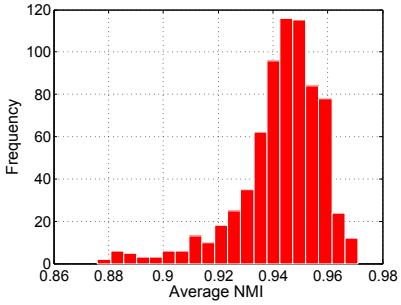

Figure 5. Average Normalized Mutual Information (NMI) of using CNM algorithm on original networks and on their compact representations as we do in our method.

Next, we measure the running time of CNM method when it is applied to each dynamic step from scratch and when it is combined with our compact representation to incrementally find dynamic modules by only considering the differences between two steps. Figure 6 shows the cumulative running time of each method as we compute the modular structures for 300 consecutive states for each patient. We see that using CNM on compressed network

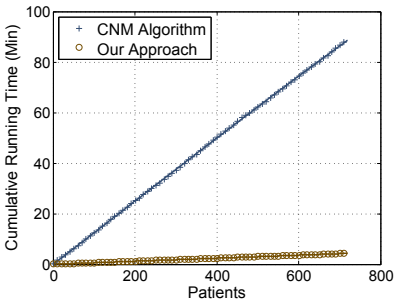

Figure 6. Cumulative running time for increasing number of patients. For legends see Figure 4.

as in our method is significantly advantageous over using it from scratch at each step on the original network. The total running time of the latter approach for all patients is more than 1.5 hours while our approach requires only a few minutes.

The gain in running time is due to the reduction in size of compact representation compared to the original network. In our experiments, the size of the compact graph (number of vertices + number of edges) is on the average 10 times smaller than the original networks (Results omitted). Therefore, our incremental method with compression uses significantly less space as well as running time. *These results imply that the improvement of running time and space utilization by using our method is reasonably large without a significant change in identified modular structure.*

IV. CONCLUSIONS

In this paper, we proposed a new approach to identify dynamic modules in BRNs. Unlike existing methods, we considered the types and directions of interactions between genes. We created a new network to represent the functional similarities of genes at a given state. In this *functional network* an edge between two genes represents the similarity of their impacts on the network state. Using this network, our algorithm identifies the modules more accurately compared to traditional methods. Additionally, our algorithm captures the dynamic behavior of BRNs as the activity levels of the genes change over time due to their interactions with each other. It incrementally updates the current modular structure to find the modular structure in the next state rather than computing it from scratch. We also benefited from the fact that the difference between two consecutive states is often very small by keeping a compact representation of the network through dynamic steps. Our experiments suggested that our method can efficiently find biologically meaningful modules that are missed by traditional approaches. Additionally, the running times showed that our approach is significantly more scalable for large size applications compared to previous approaches.

ACKNOWLEDGMENT

This work was supported partially by NSF under grants CCF-0829867 and IIS-0845439.

REFERENCES

[1] H. Ma, X. Zhao, Y. Yuan, and A. Zeng, "Decomposition of metabolic network into functional modules based on the global connectivity structure of reaction graph." *Bioinformatics*, vol. 20, no. 12, pp. 1870–6, 2004.

[2] J. Chen and B. Yuan, "Detecting functional modules in the yeast protein-protein interaction network." *Bioinformatics*, vol. 22, no. 18, pp. 2283–90, 2006.

[3] C. Wang, C. Ding, Q. Yang, and S. Holbrook, "Consistent dissection of protein interaction network by combining global and local metrics." *Genome Biology*, vol. 8(12), p. R271, 2007.

[4] E. Zotenko, K. Guimaraes, R. Jothi, and T. Przytycka, "Decomposition of overlapping protein complexes: a graph theoretical method for analyzing static and dynamic protein associations," *BMC Algorithms for Molecular Biology*, vol. 1, no. 7, 2006.

[5] J. Hallinan, "Cluster analysis of the p53 genetic regulatory network: topology and biology." in *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, vol. 7. IEEE, 2004, pp. 1–8.

[6] Y. Diao, M. Li, Z. Feng, J. Yin, and Y. Pan, "The community structure of human cellular signaling network." *Journal of Theoretical Biology*, vol. 247, no. 4, pp. 608–15, 2007.

[7] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa, "KEGG: Kyoto Encyclopedia of Genes and Genomes." *Nucleic Acids Research*, vol. 27, no. 1, pp. 29–34, 1999.

[8] T. Berger-Wolf and J. Saia, "A framework for analysis of dynamic social networks." in *International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, 2006, pp. 523–8.

[9] B. Yang and D. Liu, "Incremental algorithm for detecting community structure in dynamic networks." in *International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 4, 2005, pp. 2284–90.

[10] G. Palla, A. Barabasi, and T. Vicsek, "Quantifying social group evolution." *Nature*, vol. 446(7136), pp. 664–7, 2007.

[11] P. Pollner, G. Palla, and T. Vicsek, "Preferential attachment of communities: the same principle, but a higher level." *Europhysics Letters*, vol. 73, p. 478, 2006.

[12] J. Sun, S. Papadimitriou, P. Yu, and C. Faloutsos, "Graphscope: Parameter-free mining of large time-evolving graphs." *Conference on Knowledge Discovery in Data (KDD)*, 2007.

[13] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of online social networks." in *Proceeding of the 12th ACM SIGKDD Conference*, 2006.

[14] M. Newman and M. Girvan, "Finding and evaluating community structure in networks." *Physical Review E*, vol. 69, no. 2, 2004.

[15] A. Clauset, M. Newman, and C. Moore, "Finding community structure in very large networks." *Physics Review E*, vol. 70, 2004.

[16] F. Ay, F. Xu, and T. Kahveci, "Scalable steady state analysis of boolean biological regulatory networks." *PLoS ONE*, vol. 4, no. 12, p. e7992, 2009.

[17] J. Laura, D. Hongyue, J. Marc, and V. Vijver, "Gene expression profiling predicts clinical outcome of breast cancer." *Nature*, vol. 415, pp. 530–6, 2002.

[18] Y. Wang, J. Klijn, and Y. Zhang, "Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer." *Lancet*, vol. 38, no. 11, pp. 1289–97, 2005.

[19] C. Desmedt, F. Piette, and S. Loi, "Strong time dependence of the 76-gene prognostic signature for node-negative breast cancer patients in the transbig multicenter independent validation series." *Clinical Cancer Research*, vol. 13, no. 11, pp. 3207–14, 2007.

[20] P. Yudi and B. Judith, "Gene expression profiling spares early breast cancer patients from adjuvant therapy: derived and validated in two population-based cohorts." *Breast Cancer Research*, vol. 7, no. 6, pp. 953–64, 2005.

[21] C. Heldin, K. Miyazono, and P. ten Dijke, "TGF-beta signalling from cell membrane to nucleus through SMAD proteins." *Nature*, vol. 390, no. 6659, pp. 465–471, 1997.

[22] G. Mathern, T. Babb, P. Micevych, C. Blanco, and J. Pretorius, "Granule cell mRNA levels for BDNF, NGF, and NT-3 correlate with neuron losses or supragranular mossy fiber sprouting in the chronically damaged and epileptic human hippocampus." *Molecular and chemical neuropathology*, vol. 30, pp. 53–76, 1997.

[23] L. Danon, J. Duch, A. Arenas, and A. Diaz-Guilera, "Comparing community structure identification." *Journal of Statistical Mechanics: Theory and Experiment (JSTAT)*, vol. 9008, 2005.