LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-PROC-235067

Identify Dynamic Network Modules with Temporal and Spatial Constraints

R. Jin, S. McCallen, C.-C. Liu, E. Almaas, X. J. Zhou

September 28, 2007

RECOMB 2008
Singapore, Singapore
March 30, 2008 through April 2, 2008

Identify Dynamic Network Modules with Temporal and Spatial Constraints

Ruoming Jin[1], Scott McCallen[1], Chun-Chi Liu[2], Eivind Almaas[3⋆], and Xianghong Jasmine Zhou[2]

[1] Department of Computer Science, Kent State University, Kent, OH, USA
[2] Program in Molecular and Computational Biology, University of Southern California, Los Angeles, CA, USA
[3] Bioscience and Biotechnology Division, Lawrence Livermore National Laboratory, Livermore, CA, USA

Abstract. Despite the rapid accumulation of systems-level biological data, understanding the dynamic nature of cellular activity remains a difficult task. The reason is that most biological data are static, or only correspond to snapshots of cellular activity. In this study, we explicitly attempt to detangle the temporal complexity of biological networks by using compilations of time-series gene expression profiling data. We define a dynamic network module to be a set of proteins satisfying two conditions: (1) they form a connected component in the protein-protein interaction (PPI) network; and (2) their expression profiles form certain structures in the temporal domain. We develop the first efficient mining algorithm to discover dynamic modules in a temporal network, as well as frequently occuring dynamic modules across many temporal networks. Using yeast as a model system, we demonstrate that the majority of the identified dynamic modules are functionally homogeneous. Additionally, many of them provide insight into the sequential ordering of molecular events in cellular systems. We further demonstrate that identifying frequent dynamic network modules can significantly increase the signal to noise separation, despite the fact that most dynamic network modules are highly condition-specific. Finally, we note that the applicability of our algorithm is not limited to the study of PPI systems, instead it is generally applicable to the combination of any type of network and time-series data.

⋆ EA's work was performed under the auspices of DoE by University of California, LLNL under Contract W-7405-Eng-48, with support from the LDRD office

1 Introduction

Cellular systems are highly dynamic and responsive to cues from the environment. Cellular function and response patterns to external stimuli are regulated by a complex web of diverse molecular interactions, such as protein-protein interactions, protein-DNA interactions, and metabolic processes. Despite the availability of large-scale biological network data [25, 18, 6, 7], gaining a system-level understanding of the *dynamic* nature of cellular activity remains a dif cult and, until recently, a much overlooked task. Since there typically is little direct information available on the temporal dynamics of these network interactions, the majority of molecular-interaction network modeling and analysis has been solely focused on static properties. For example, a protein-protein interaction (PPI) network obtained from yeast two-hybrid experiments can be viewed as a comprehensive graph of the edges (interactions) that eventually may occur under the set of tested conditions. However, it is not guaranteed that two adjecent edges would ever occur *simultaneously*, and thus, that identi ed network modules and motifs will correspond to functionally relevant units. While a series of studies have been aimed at identifying modules in PPI networks [21, 2, 23], the networks in their analysis have all been regarded as static.

In this study, we attempt to detangle the *temporal* complexity of biological networks by identifying dynamic modules, consisting of a set of proteins that coexist both spatially and temporally. We will use the PPI network as the spatial constraints. We estimate temporal characteristics of a network component by using compilations of time-series gene expression pro ling data, since accurate temporal parameters are not yet available for PPI systems. A similar approach has been employed to study the dynamic protein complex formation during the cell cycle [3, 15]. In order to measure the similarity between two time-series gene-expression pro les [1, 22], we have implemented time-warping dynamic programming. A signi cant advantage of this method is that it identi es the best local alignment of segments in two time-series pro les. Consequently, it can identify time-shifted and local similarity patterns which are often overlooked by computing the Pearson's correlation of two entire pro les. We represent the local alignment of two pro les as a range pair (or interval pair), denoted as $[s1, e1]$:$[s2, e2]$, where $s1$ and $s2$ are the starting time points, and $e1$ and $e2$ are the end time points of the alignment. Since two proteins may interact over different time segments in a long time-series and they may show varying level of cooperativity in those time segments, we may also consider suboptimal alignments of two pro les. In this case, the temporal expression similarity of two interacted proteins can be represented as a list of range pairs. Given a PPI network, we add the local alignment information from the time-series gene expression data, and refer to the network as a *temporal network.*

On a temporal network, we can discover how two proteins' activities correlate with each other over time. More importantly, for multiple proteins we can identify when a dynamic module is activated in the PPI network. We de ne a dynamic network module to be a set of proteins which satisfy two conditions: (1) they form a connected component in the PPI network; and (2) their expression pro les form certain structures (see below) in the temporal domain. An example dynamic module satisfying these constraints could consist of all proteins which time-series local alignments overlap over a common time period. That is, all protein interactions in this example module are synchronized to perform their functions. Another interesting example corresponds to the case of activity cascades. Here, a dynamic module is triggered by one or a small number of proteins that activate their network neighbors. These in turn, may also propogate the signal further through neighbor activation. We introduce an *event model* to describe the temporal relationship among proteins that participate in such activation cascades. Simply stated, we use a directed edge to represent how the activity of a protein may invoke that of another protein, and the direction is determined by the range-pair of their time series alignment, i.e. the beginning aligned time point of the triggering protein is likely to be earlier than that of the triggered protein.

Given the rapid increase in time-series microarray data available in the public domain, we have further extended our dynamic network analysis to uncover network modules occuring frequently under *different* time-series conditions. This approach provides two advantages: Since noise is a common problem in all high-throughput data (false-positive or false-negative determination of interactions), we can enhance the signal to noise separation by leveraging our identi ed, frequenly occurring signals). Also, by combining information about the experimental conditions associated with each time series data, *e.g.* heat-shock, acid shock or gene disruption, we can explore the condition-speci city of dynamic modules.

Although many graph algorithms are available to perform network analysis, to our knowledge, no algorithm has been developed to identify network patterns while accounting for temporal constraints. In this study, we take the rst step in this direction by developing ef cient mining algorithms to discover dynamic modules in a temporal biological network, as well as frequent dynamic modules across multiple temporal networks. We have performed detailed experimental testing while using yeast as a model system. Our results show that a large portion of the identi ed dynamic motifs are functionally homogeneous. In addition, many of those modules provide insight into the sequential order of molecular events in cellular systems. Furthermore, a majority of the discovered modules are not densely connected, yet they comprise functional units. Such modules are very dif cult to identify based on the PPI network alone. Finally, we have demonstrated that identifying frequent dynamic network modules can signi cantly enhance the signal to noise separation, despite the fact that most dynamic network modules are highly condition-speci c. We are well aware of the limitation of microarray data in measuring protein activities, and the incompleteness of protein-protein interaction data. While using those data as an example, we want to emphasize that our algorithm is generally applicable to combining any types of network and time-series data.

2 Problem Definition

We represent the PPI network as a graph $G = (V, E)$, where proteins correspond to the vertex set V, and the protein-protein interactions are recorded as the edge set E. Each vertex (protein) is associated with a gene expression time-series, formally $T_i = (x_1^i, x_2^i, \cdots, x_n^i)$. The *temporal* network is then the graph G combined with the time series similarity information between interacting proteins.

Here, we use the time-warping dynamic programming algorithm [1, 22] to align two time-series expression pro les, and use a list of range pairs to represent the resulting alignments for each interacted protein pair. The range pair list is denoted as $\{[i_1, i_1'] : [j_1, j_1'], [i_2, i_2'] : [j_2, j_2'] \cdots [i_k, i_k'] : [j_k, j_k'], \cdots\}$, where i_l, j_l are the starting points of the l-th aligned interval pair of an interacted protein pair, and i_l', j_l' are the end points. Consequently, the time series subsequence between i_l and i_l' of one protein is very similar to the time series subsequence between j_l and j_l' of the other protein. Given this, we use $G^R = (V, E, R)$ to denote the temporal biological network, where R records the list of similar range pairs for each edge. In addition, we drop the edges (the interacted protein pairs) from E if they do not have any local similar range pairs. We will now focus on performing the dynamic module discovery on the temporal biological network.

2.1 Dynamic Module Discovery

As previously mentioned, the module discovery approach in this study is bound by both *spatial* and *temporal* constraints. Here, the spatial constraints refer to the physical interactions between proteins. In terms of temporal constraints, for each vertex in the module, we require that the time range pairs associated with all adjacent edges to this vertex shall share a common time interval of minimal length d. Note that our de nition of temporal constraints is not unique. For instance, we may require all the edges in the modules to share a common time interval, or we may require that all the time intervals appear within a speci ed time period. Due to space limitation, we will only focus on the rst variant. However, the techniques we have developed can easily be modi ed and applied to a wide variety of alternative de nitions of temporal constraints. Given this, we formally de ne the *dynamic module* as follows.

Definition 1 *Given a temporal network $G^R = (V, E, R)$, a dynamic module is an induced and connected subgraph $G[S]$ for a vertex set $S \subseteq V$ with a range function r, which assigns each edge $e = (v_i, v_j) \in E(G[S])$ with a range pair $r(e) = (r(e)[v_i] = [y_i, y_i'] : r(e)[v_j] = [y_j, y_j']) \in R(e)$, and all the intervals specified by r satisfy the following temporal constraints: $\forall v \in S$, $|\cap_{e \in E(G[S]) \wedge v \in e} r(e)[v]| \geq d$. The dynamic module is denoted as $G^r[S]$.*

Figure 1(a) lists an example of a dynamic module with the choice of temporal constraint parameter $d = 2$. Note that the range pairs on the outside of the subgraphs represent the local similarity alignment, while the ranges on the inside of the subgraphs represent the intersection of those alignments.

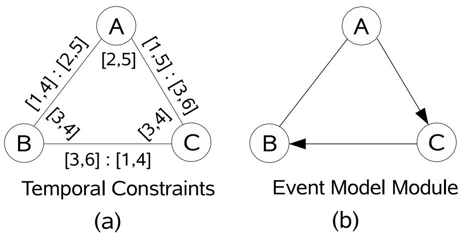

Fig. 1. A Dynamic Module and its Event Model

2.2 Event Model and Frequent Dynamic Module

The range pairs in the dynamic module provide the sequential ordering of activities between edge-connected vertices. Speci cally, if one interval in the range pair appears earlier than the other one, it could be possible that the activity of the vertex (protein) associated with the rst interval induces the activity of the other vertex. That is, the sequential order obtained could facilitate causal inference, although causal inference requires much more information and is beyond the scope of this work. In the following we will use a directed edge to represent the sequential order between two vertices in the dynamic module. Speci cally, let $[t_{A1}, t_{A2}] : [t_{B1}, t_{B2}]$ be the interval pair between protein A and B. If $t_{A1} - t_{B1} \geq k$, we build a directed edge from protein A to B; if $|t_{A1} - t_{B1}| < k$, we build an undirected edge between A and B, where k serves as a smoothing parameter against noise. This new representation of the dynamic module is referred to as the *event model* module. For a dynamic module $G^r[S]$, we denote its corresponding event model module as $G^e[S]$. Figure 1 (b) shows the event model for the 3-vertex dynamic module of Figure 1(a).

Next we introduce the concept of the *frequent dynamic module* in a biological network with multiple (microarray) time course data. In other words, for a given PPI network $G = (V, E)$, we can have several different range functions, such as R_1, R_2, $\cdots$, R_m, each extracted from different time course datasets. For each temporal network G^{R_i} and a given set of vertices, we can then produce a set of dynamic modules, $G^{r'_1}[S]$, $\cdots$, $G^{r'_k}[S]$. We are interested in uncovering substructures appearing frequently in these dynamic modules, that is, dynamic events frequently occurring in multiple, different time-course datasets.

Definition 2 Frequent Dynamic Motif: *Given a support θ and a PPI network graph $G = (V, E, R)$ with m range-pair functions, $R_1, \cdots, R_m$ from m time course datasets, and a set of vertices $S \subseteq V$. Let $G^{r'_1}[S]$, $\cdots, G^{r'_k}[S]$ be the dynamic modules occurring in $G^{R_1}, \cdots, G^{R_m}$, and $G^{e'_1}[S], \cdots, G^{e'_k}[S]$ be the corresponding event motif modules. A frequent dynamic module is thus defined to be the subgraph which connects all the vertices in S and appears in at least θ event-model modules out of the k possible.*

In addition, in order to reduce the number of frequent dynamic modules, we introduce the *maximally frequent dynamic modules* with respect to the vertex set S: a frequent dynamic modules G_S of S is *maximally frequent* if no other frequent dynamic modules of S contain G_S as a subgraph.

3 Algorithms

3.1 Algorithm for Dynamic Module Discovery

Given a temporal network $G^R = (V, E, R)$, we aim to ef ciently discover all the dynamic modules in a temporal biological network. First, we note a couple of interesting properties of this problem: (1) The underlying topology of the dynamic module is a connected induced subgraph $G[S]$, where S is vertex set. (2) Since each edge in the temporal network is associated with a list of range pairs, and each edge in the dynamic module only has one range pair, it is straightforward that $G[S]$ may serve as the underlying topology of multiple dynamic modules. Hence, one may simply choose a different range pair over each edge for the same induced subgraph in order to generate a different dynamic module. For instance, for the case of a 3-clique where each edge contains a list of L range pairs, we may have a maximum of L^3 dynamic modules. Consequently, a major challenge of dynamic module discovery is to ef ciently handle the potential multitude of edge-range combinations, as well as the spatial and temporal constraints. We will refer to all dynamic modules sharing the same underlying topology ($G[S]$) as the dynamic module group.

The central idea of our algorithm is as follows: We simultaneously discover all dynamic modules with the same underlying topology (the dynamic module group) by starting from a single vertex. We recursively enumerate the dynamic module groups in a depth- rst fashion, and recursively append new vertices to the identi ed dynamic module groups. In order to ef ciently implement this approach, we have to speci cally consider these two challenges: 1) We need to devise an ef cient scheme for handling the dynamic module groups, as their size can be very large. For instance, in the 3-clique example, the size of the dynamic module is L^3, and explicitly keeping track of all the combinations can quickly become computationally expensive. 2) It is necessary to enumerate all dynamic module groups completely and without repetition in an ef cient manner. Note that this challenge is similar to that of enumerating all connected, induced subgraphs.

In the next section, we will introduce the two key techniques of our mining algorithm that are central to managing the temporal and spatial constraints. We will   rst focus on resolving the temporal constraints by introducing a compressed representation of the dynamic module group. Then we focus on enumeration ordering, which not only will resolve the spatial constraints, but also avoid any redundancy.

Managing Temporal Constraints: The naive (and simple) approach to manage dynamic module groups is to record each dynamic module individually, and each dynamic module records a range pair for each edge. Unfortunately, this is likely rather computationally expensive, even when the lists of the range pairs over each edge on the network is very small: Consider the example of a 10-clique, where each edge only has 2 range pairs. For this case, the possible number of dynamic modules is 2^{45}, and each module records $\binom{10}{2} = 45$ pairs.

Instead, we propose a novel representation for a dynamic module group, the vertex range list, which signi cantly reduces memory and update costs for each dynamic module group. Intuitively, for each node, we record all common intersection intervals from its adjacent range-pair list: If a node has degree k, the common interval is the intersection of any k intervals, each selected from the range pair list over the k adjacent edges. Formally, for the induced subgraph $G[S]$, we de ne $R(e)[v], v \in e$ to be all the intervals associated with v and the edge e, where $R(e)$ is the list of range pairs over e. Furthermore, we de ne the join operator $\otimes$ over two interval list (X and Y) as follows:

$$X \otimes Y = \{z | z = x \cap y, x \in X \wedge y \in Y, |z| \geq d\}$$

where x and y are intervals and their intersection is the interval z, and d is the temporal constraint parameter (the common intersection is longer than d). Hence, for each node, we record $R(v) = \otimes_{e \in E(G[S]) \wedge v \in e} R(e)[v]$. Thus, the total memory cost for recording the dynamic module group of $G[S]$ is $O(\sum_{v \in S} |R(v)|)$. Using this approach on our example of the 10-clique, we would only need to record a maximum of 10×2^9 intervals for the dynamic module group.

An additional strength of this approach is that the vertex range list can be generated incrementally for the dynamic module: Let $S.R$ be the vertex range list representation for the existing dynamic module group of S and let Q_s be a set of new vertices. The simple procedure of $Join(Q_s, S)$ in Algorithm 1 can easily produce the representation for the dynamic model group of $Q_s \cup S$.

The vertex range-list representation also facilitates a swift validation of the temporal constraints by the following theorem.

Theorem 1 *For a candidate dynamic module group of $G[S]$, if one of its vertex $v \in S$ has $R(v) = \emptyset$, then no dynamic module in the group satisfies the temporal constraints.*

Finally, we note that extracting individual dynamic modules from the group is straightforward. We start by recursively selecting each intersection interval from every node and color the related edge using the range pair containing the intersection interval. The dynamic module is then produced by edges in a subgraph that are colored more than once. Unfortunately, we have to omit further details due to space constraints.

Maintaining Spatial Constraints and Enumeration Ordering: To maintain the spatial constraints, we incrementally expand the underlying topology (the induced subgraph) through its neighbor vertices. Hence, we have to be careful to avoid redundant enumeration. Note that this problem description can be transformed to that of enumerating the connected, induced subgraphs.

Our strategy is as follows. First, we partition the vertex sets into different non-overlapped groups. Assuming the vertex set of the network G is $V(G) = \{v_1, v_2, v_3, \cdots, \}$, the rst group will include all connected vertex sets containing v_1; the second group includes all connected vertex sets containing v_2 excluding v_1; and the third group includes all connected vertex set containing v_3 while excluding v_1 and v_2. As a result, two vertex sets in different groups will not be redundant. Note that each connected vertex set S corresponds to a connected, induced subgraph $G[S]$.

Second, we enumerate the connected vertex set within each group. We achieve this by transforming the graph G into a tree rooted by the current working vertex v_i. The h-level of the tree contains the vertex in the graph G which has the shortest path reaching v_i with length h. Clearly, to build this tree, we only need to perform a Breadth-First Search (BFS) of the graph starting from v_i. Once the tree has been constructed, we enumerate all connected vertex sets containing v_i in the following way: We start from the vertex set containing only v_i, which is in the rst level of the tree ($\{v_i\} = T[1]$). Second, any valid vertex set in the next level of the tree must satisfy the condition of being the subset of the neighbors of the newly added vertex (ensuring that connectivity of the subgraph is maintained) which appears in the next level of the tree. Hence, we recursively append new vertices to the current vertex set. Note that the spatial constraints are maintained for this append operation. It is straightforward to prove the completeness and non-redundancy (that no connected vertex set is enumerated more than once) of this enumeration.

An outline of the complete discovery algorithm is given in Algorithm 1.

Optimization: We introduce two optimization techniques to reduce the mining cost. The rst opti-

Algorithm 1 $DynamicModuleMiner(Graph\ G)$

```
1: for each $v \in V(G)$ do
2:     $T \leftarrow GenerateBFSTree(G, v)$; {Generate the Breadth First Search Tree rooted by $v$;}
3:     $S \leftarrow \{v\}$; {All the induced subgraph contains $v$ as its first node}
4:     RecursiveDMMine($S$, $\{v\}$, $T$, 1); {Enumerate all the induced subgraph containing $v$}
5:     $G \leftarrow G - v$; {$v$ is removed from $G$}
6: end for
Procedure $RecursiveDMMine(S, N, T, h)$ {S: the vertex set of the induced subgraph;} {T: the BFS tree rooted by $v$;} {N: the vertex set
    just added into S;} {h: the level of the tree currently being visited;}
7: $Q \leftarrow Neighbor(N) \cap T[h+1]$; {The possible vertex set which can be added to the current subgraph}
8: for each $Q_s \subseteq Q$ do
9:     $S' \leftarrow Join(Q_s, S)$; {Adding vertex set $Q_s$ to S}
10:    if $S' \neq \emptyset$ then
11:        $Empty \leftarrow Output(S')$; {Extract all dynamic modules from the dynamic module group recorded by the range list representation}
           {Empty=TRUE: No dynamic modules}
12:        if NOT $Empty$ AND $h+1 < T.height$ then
13:            RecursiveDMMine($S', Q_s, T, h+1$);
14:        end if
15:    end if
16: end for
Procedure $Join(Q_s, S)$
17: $S' \leftarrow Q_s \cup S$;
18: $\forall s \in S : S'.R(s) \leftarrow S.R(s)$ {copy the range list from $S$ to $S'$}
19: $\forall v_1 \in Q_s : S'.R(v_1) \leftarrow \emptyset$ {initialize the range to the empty set}
20: for each $e = (v_1, v_2) \in E(G) : v_1 \in Q_s \wedge v_2 \in S$ do
21:    $S'.R(v_1) \leftarrow S'.R(v_1) \otimes R(e)[v_1]$; {$R(e)[v_1]$:all the intervals associating with $v_1$ on the edge $e$}
       { Let $\emptyset \otimes R(e)[v_1] = R(e)[v_1]$ for initialization of $v_1$ range list}
22:    $S'.R(v_2) \leftarrow S'.R(v_2) \otimes R(e)[v_2]$; {$R(e)[v_2]$:all the intervals associating with $v_2$ on the edge $e$}
23:    if $S'.R(v_1) = \emptyset$ OR $S'.R(v_2) = \emptyset$ then
24:        return $\emptyset$; {Theorem 1}
25:    end if
26: end for
27: return $S'$;
```

mization step is to reduce the cost of building the BFS tree. Note that the cost of building the BFS tree is essentially determined by the size of the connected components containing v. To reduce the total running cost, we would like to remove the vertices which maximally reduce the size of largest connected components. The following heuristic will achieve this goal: when choosing a vertex at each iteration, we always rst pick the vertex with the maximal degree. This heuristic has previously been used in a complex-network study to break the large graph into pieces [17].

Another major cost of the algorithm is associated with choosing a valid vertex set to append to the existing set of vertices (Line 7). This step in the algorithm requires a set intersection operation, which can be very expensive when the sets are large. We optimize this operation by removing the explicit intersection operation as follows: We rst mark each vertex in the graph with its level of the BFS tree, i.e. the length of its shorted path to the current root node. Second, we identify all the neighbors of vertex set N which are marked by $h+1$ in the $Neighbor(N)$ procedure. Taken together, these two steps implement the intersection operation in Line 7: $Neighbor(N) \cap T[h+1]$.

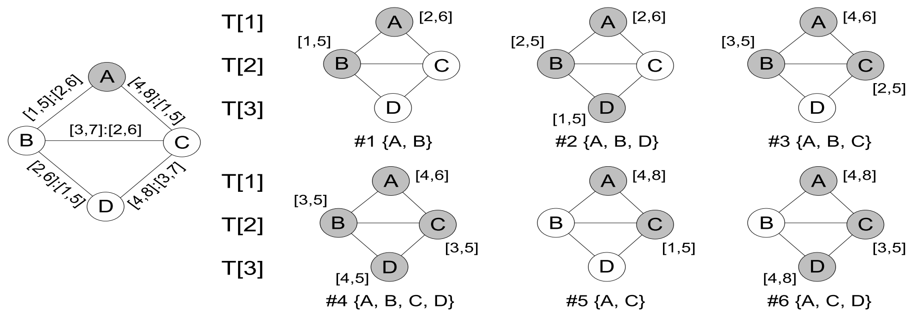

Fig. 2. Dynamic Module Discovery

Running Example: In Figure 2 we show an example of the dynamic module discovery process. For the sake of simplicity, we only associate each edge with a single range pair. With the minimum intersection length set to $d = 2$, we perform the discovery as follows. Initially the module $\{A, B\}$ allows both vertices to maintain their respective intervals. Adding D causes the local intersection of the intervals at B, changing B's interval to $[2, 5]$. Next we look at the $\{A, B, C\}$ module in which all three local intervals change due to the addition of C to $\{A, B\}$. Until this point, all local intervals satis ed the minimum intersection criterion of $d = 2$. While adding D during the next iteration, we see that the intersection of D with edges (B, D) and (C, D) drops below the minimum length requirement. Consequently, module $\{A, B, C, D\}$ is not recorded in the nal result. However, when skipping to the last step, we see that module $\{A, C, D\}$ is recorded in the result as all of its local intersections meet the speci ed requirements. Note that our appending procedure follows the breadth- rst tree created from the root A, and $T[1], T[2]$ and $T[3]$ are the levels of the tree.

3.2 Algorithm for Frequent Motif Discovery

Algorithm 2 $FrequentDynamicModules(D, h, k)$

```
Parameter: $D$, a set of hash tables, each hash table holds all the dynamic modules for a dataset
Parameter: $h$, the working hash table
Parameter: $k$, the number of combined hash table
Parameter: $\theta$, the minimum support level
    {Invoked by $FrequentDynamicModules(D, \emptyset, 0)$, $D$: all the input hash tables}
 1: if $k \geq \theta$ then
 2:     return; {only enumerate the maximal frequent modules}
 3: end if
 4: for each $d \in D$ do
 5:     $D \leftarrow D \setminus d$
 6:     $h' \leftarrow intersect(d, h)$; {interesect the dynamic modules with a previously unseen hash and $d = intersect(d, \emptyset)$ for initialization}
 7:     if $h' \neq \emptyset$ $AND$ $k = \theta$ then
 8:         $result \leftarrow result \cup h'$; {record the maximal frequent modules}
 9:     end if
10:     if $h' \neq \emptyset$ then
11:         $FrequentDynamicModules(D, h', k+1)$;
12:     end if
13: end for
```

While several graph mining algorithms have been developed to identify frequent subgraphs [13, 28, 11, 29], the majority of existing approaches either perform level-wise or a depth- rst-search enumeration of pattern space. However, these approaches are incommensurate with the structure of our problem. First, our inputs are different from these algorithms: we have already discovered all dynamic modules for each time series dataset. Second, to reduce the number of frequent dynamic modules, we aim to discover all the maximally frequent dynamic modules with respect to each vertex set. In addition, we note that the number of time series datasets is generally not large.

Taking these factors into account, we develop an ef ciently mining algorithm (shown in Algorithm 2) to discover all maximally frequent dynamic modules. We store all dynamic modules resulting from each individual time series dataset in a hash table. Note that each dynamic module is represented as a directed graph, i.e. the event model dynamic module. The set of the hash table is recored as D in Algorithm 2. Since we load each table during each iteration (Line 4), we do not have to simultaneously load the entire D into the main memory. In the subsequent steps, we only need to maintain a working hash table h' to record the common subgraphs for the vertex sets.

Initially, h' will store all the dynamic modules for one dataset. Then, when selecting another hash table d (which records all the dynamic modules for another dataset), we will intersect h' and d (Line 6). The intersection operation is performed in three steps: 1) we locate the dynamic modules with common vertex set; 2) we generate the common subgraph by intersecting the corresponding edges of the selected dynamic modules, i.e. an edge is accepted in the common subgraph if both original dynamic modules contain the edge; 3) we evaluate the connectedness of the common subgraph. If the resulting subgraph is disconnected, it is discarded.

In essence, the recursive procedure in Algorithm 2 will select all the θ datasets from the total dataset before intersecting the corresponding hash tables. Additionally, we test for containing relationships when we combine the output of each interaction into the nal result set $result$, i.e. if one dynamic module is isomorphic to, or a subgraph of, another dynamic module in the result set, it is discarded. In this way, the nal result set $result$ will have all the maximally frequent dynamic modules with respect to each vertex set.

4 Results

4.1 Data source and alignment of time-series expression data

We collected 36066 distinct protein-protein interactions of *S. cerevisiae* from The BioGrid databases [24], the MIPS database [8], and other sources [27, 12]. As is customary, self interactions representing autoregulation or protein homodimerization were not included in the analysis. We collected a total of 11 microarray gene expression time-series for *S. cerevisiae*. These datasets include from 10 to 25 time points of various temporal scales (*e.g.* minutes and hours) and measure gene-regulation under very different biological conditions, such as the cell cycle, lamentous-form growth, fermentation, carbon source perturbation, and thiolutin treatment.

To analyze these time-series, we implemented local similarity analysis [22], which is a variant of the time-warping algorithm, to identify the local alignment between two time series. Since most of the collected time-series expression datasets are short, here we only identify the best local alignment for each protein pair and do not consider suboptimal alignments. For each gene pair with an identi ed protein-protein interaction, we performed a normal score transformation [14] for their expression pro les before analyzing their local alignment by dynamic programming. We de ne the local similarity score as the maximal sum of the product of the corresponding entries of all the subsequences of the two time-series within a prede ned time delay D. Hence, the value of D indicates how far apart in time the gene-gene interactions are allowed to occur. We choose $D = 5$ in our analysis. To determine the signi cance threshold ($P < 0.025$) of the computed local similarity score, we carried out the local similarity analysis on 1,000,000 pairs of random expression pro les with a normal distribution. We assigned an edge between all gene pairs with a known protein interaction if we also uncovered a signi cant local expression similarity score between the two genes.

From protein interaction data and the 11 expression time series, we obtain 11 graphs, ranging from 1872 to 13298 edges. Interestingly, the terminal genes of 32% of those edges have Pearson's correlations with

P-value higher than 0.05. Consequently, the local alignments are either shifted or are short for all of these edges and, thus, can not be captured by standard correlation analysis.

4.2 Dynamic network modules are more biologically meaningful than static modules

We have obtained all dynamic network modules with different overlap parameter $d = 3, 4, 5, 6, 7, 8$. Using the cell cycle dataset (GDS2347) as an example, at $d = 8$, we identi ed 4744 modules of node size 5 to 10. To assess whether the temporal information improves our ability to discover biologically meaningful modules, we compared the functional homogeneity of the dynamic modules with that of the static modules determined based only on the protein interaction data. That is, for each of the dynamic modules, we identify the connected component of the PPI network with the same size, and compare the homogeneity of their biological functions. We used the Gene Ontology (GO) biological process annotation and de ned speci c functions to be those associated with GO nodes containing less than 200 genes. A module is considered functionally homogeneous if its variation in functional classi cation, as modeled by the hypergeometric distribution, has a $P < 10^{-6}$ (P stands for P-value). Our results showed that 46% of dynamic modules are functionally homogenous, while on average (from 100 randomizations), only 35% of static modules are functionally homogenous. This highlights the usage of temporal information in uncovering biologically meaningful gene constellations that are hidden by a static network representation. In general, the functional homogeneity of modules increases with increasing d. Additionally, 31% of the dynamic modules are statistically enriched in proteins from the same protein complexes ($P < 10^{-3}$), while this is on average true only for 25% of the static modules. The functional analysis results of all datasets (at $d = 8$) are listed in Table 1.

Interestingly, a majority of the dynamic module are very sparse. For example, 97% of the dynamic modules from the dataset GDS2347 have a connectivity γ less than 0.5, where γ is de ned as $\gamma = 2m/(n(n-1))$, with m being the number of edges and n the number of nodes in a module. Strikingly, if analyzed in conjunction with the original protein-protein interaction network, nodes in 88% of the dynamic modules are loosely connected with density $\gamma < 0.5$. We have illustrated two such examples in Figure 3. The non-dense modules are very hard to extract from the static protein-protein interaction network solely based on their topology. Here, we demonstrate that the temporal expression information can facilitate the identi cation of likely important signals that are otherwise easily overlooked.

Table 1. GO functional homogeneity in dynamic network modules

Dataset	# modules	GO[a]	Complex[b]
GDS124	17783	51.1%	38.2%
GDS1611-1	24217	42.0%	31.4%
GDS1611-2	33111	38.7%	29.4%
GDS1752-1	17684	40.5%	26.1%
GDS1752-2	16170	40.3%	29.2%
GDS18	23052	42.9%	34.9%
GDS2318	12181	48.9%	30.0%
GDS2347	8053	54.9%	30.1%
GDS2350	4744	42.2%	22.5%
GDS39	16385	43.7%	32.5%
GDS608	1954	61.1%	38.5%

[a] The percentage of modules with significant enrichment in proteins from a GO term ($P < 10^{-6}$ and required that at least 2 genes in the module fall in the same GO term)
[b] The percentage of modules with significant enrichment in proteins from a protein complex ($P < 10^{-3}$ and required that at least 2 genes in the module belong to the same protein complex).

4.3 Dynamic network motifs reveal temporal events in cellular systems

The dynamic network modules can aid in the characterization of signal propagation and the elucidation of temporal organizational patterns in cellular activities. Note that, there is little information on the detailed sequential ordering of molecular events due to the limited power and resolution of current experimental techniques. This leaves an important opportunity for the community of computational scientists, as *in silico* predictions can complement experimental approaches to provide novel insights. In the following, we will discuss two cases from our analysis in detail, for which the identi ed dynamic events are supported by current knowledge.

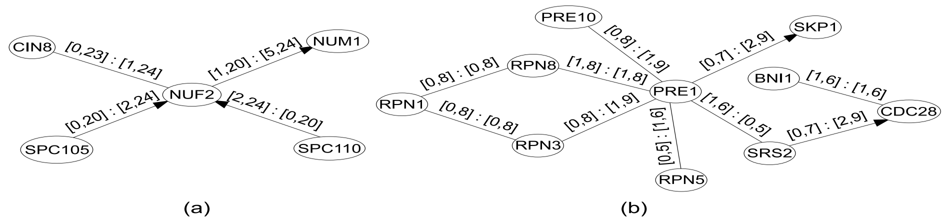

Fig. 3. Two examples of dynamic network modules.

Figure 3(a) illustrates a dynamic network module activated during the cell cycle progression (obtained using dataset GDS2350). The module contains 5 genes, all of which are involved in the biological process of microtubule cytoskeleton organization and biogenesis . Among those, SPC105 and SPC110 are components of the spindle pole body; NUF2 is implicated in connecting the centromere to the spindle pole body during chromosome segregation [16]; Cin8 clusters kinetochores into their characteristic bi-lobed metaphase con guration [26]; while NUM1 is required for nuclear migration and localizes to the mother cell cortex as well as the bud tip, forming cytoplasmic microtubule capture-sites during the late anaphase [4, 5]. Although all of these genes participate in microtubule dynamics, their interactions do not occur at the same time. The directed edges from SPC105/SPC110 to NUF2 indicates that the assembly of the spindle pole body takes place ahead of its interaction with the centrosome, and the directed edge from NUF2 to NUM1 implies that the connection from the centromere to the spindle pole occurs before nuclear migration. The identi ed dynamic network module correctly captures the sequential order of protein-protein interactions along the time axis. It is noteworthy that all of the 4 connected gene pairs have the very low Pearson's expression correlations of -0.34, -0.08, -0.12, and 0.35. Consequently, this module can only be identi ed by taking the temporal characteristics into account.

Our second example, Figure 3(b), shows a dynamic module activated in the dataset GDS608 [19]. This dataset resulted from studies of the temporal expression patterns in wild-type diploid cells shifted from the typical yeast-form growth to that of a lamentous-form growth. Filamentous-form cells were collected and pro led hourly for 10 hours. The identi ed dynamic module contains 10 genes, in which RPN1, RPN3, RPN8, PRE1, PRE10, PRE5 are involved in ubiquitin-dependent protein catabolic process. In particular, PRE10 and PRE5 belongs to the proteasome alpha-subunit complex, RPN3 and RPN8 are part of a proteasome regulatory particle, the lid subcomplex, while SRS, SKP1, CDC28, and BNI1 are involved in the cell cycle. Most edges in this module are undirected (that is, they contain at most a time shift of 1 unit), and only two edges are directed, one pointing from PRE1 to SKP1, and the other pointing from SRS2 to CDC28. This is in agreement with the hypothesis that the ubiquitin-dependent protein degradation by the 26S proteasome is controlling the lamentous-form growth [19]. More speci cally, the 26S proteasome regulates the activity of the Cdc28 kinase, which in turn controls cell-cycle progression and morphogenesis during lamentous-form growth [19].

4.4 Identify frequent dynamic network modules across different conditions

Based on the dynamic network modules derived from 11 conditions (module sizes range from 5 to 10 nodes with $d = 8$), we identi ed modules recurring in at least 2 different conditions. The percentage of functionally homogenous modules increases signi cantly with the module recurrence frequency (Table 2), demonstrating the utility of using recurrence to lter out noise and rarely occuring modules. Interestingly, at a recurrence level of $\geq$ 3, almost none of the frequent network modules contain directed edges. Note that undirected edges represent approximate co-expression relationships, and that directed edges represent time-shifted expression similarity, highlighting the complexity of sequential events in the cellular systems. That is, most of the sequential events are highly speci c to a particular intrinsic condition or external perturbation. This is further con rmed by the fact that the majority of recurrent modules with directed edges are activated under similar conditions, such as cell cycle pro ling of wild-type or mutant yeast, or carbon source perturbation at different levels of severity. Furthermore, frequently occurring modules exhibit signi cant house-keeping characteristics: at recurrence 6, 35% of the patterns are related to ribosome biogenesis and assembly .

Table 2. GO functional homogeneity in frequent dynamic network modules

Support [a]	# Frequent modules	Enrichment percentage[b]
2	5822	88.5%
3	2059	96.9%
4	745	98.4%
5	183	99.5%
6	22	100.0%

[a] The number of datasets that support the frequent occurrences.
[b] The percentage of modules having significant enrichment with a GO term ($P < 10^{-6}$ and required that at least 2 genes in the module fall in the same GO term).

In the following, we discuss two examples of recurrent network modules. In Figure 4(a), a four-gene module is activated under two different types of experiments: (1) analysis of wild-type and the *upf1*-mutant strains after treatment with 10 μg/ml thiolutin, a global transcription inhibitor; and (2) analysis of steady state cultures growing on galactose following a pulse of 2 grams of glucose per liter. Three out of those four genes, TRM11, TRM9, TRM112, are known to be involved in tRNA methylation. The fourth gene, Lys9p, is a Rossmann-fold protein that may utilize NAD/NADP as a cofactor and could play a negative regulatory role on several tRNA modi cation activities through its interaction with Trm112p [9, 20]. In Figure 4(b), a three-gene module GAL3, GAL8, and GAL1 appear under condition (2) but with the addition of only 0.2 grams of glucose. All three genes are involved in the positive regulation of transcription by galactose. The interaction between GAL3p and GAL80p activates the ef cient utilization of galactose, and GAL1p is a key enzyme in galactose metabolism [10]. In the presence of glucose, these genes are rapidly and fully repressed [10], and, being signaling molecules, GAL3 and GAL8 are expected to show a more rapid response than the metabolic enzyme GAL1p. Note that dynamic modules derived here only re ect potential sequential events, and do not imply causal effect. However, extracting sequential observations is often the rst step in causal inference.

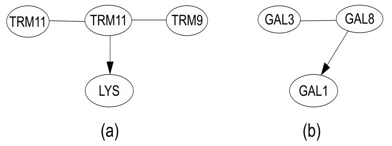

Fig. 4. Two examples of recurring dynamic network modules.

5 Conclusions

We have presented the rst graph-based algorithm to identify dynamic network building-blocks by combining information from multiple temporal networks. To this end, we have designed a set of ef cient algorithms to identify *dynamic* network modules. Our approach provides an alternative to traditional graph-based module- nding algorithms that assume all links are simultaneously present (static). Consequently, traditional algorithms are incapable of addressing questions such as the time-ordering of link-based events.

In contrast, our approach facilitates the use of temporal information to detangle the complex wiring diagrams of cellular systems. As an example, we have applied our algorithm to a combination of microarray time-series expression data and the yeast protein-interaction network. We have demonstrated that our method uncovers functionally homogenous network modules, and more importantly, that it has the ability to identify the *sequential ordering* of molecular events taking place in biocehmical systems. While the results from our approach have no direct implication on the causality of events, the sequential activity information embedded in the derived modules may serve as a starting point for causal-inference analyses.

Microarray gene-expression data and protein interaction data have their speci c limitations, and as such, may not provide a measurement of all functional activities in a cell. However, we want to emphasize that as a general data integration framework, our algorithm can be applied to any type of time-series activity measurement (not limited to gene-expression data) and any type of network data (not limited to protein-interaction data). Thus, our approach provides an important new class of tools for the systems-level analysis of biological data.

References

1. J Aach and G M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, Jun 2001. Comparative Study.
2. Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, Guojie Li, and Runsheng Chen. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Res*, 31(9):2443–2450, May 2003. Comparative Study.
3. Ulrik de Lichtenberg, Lars Juhl Jensen, Soren Brunak, and Peer Bork. Dynamic complex formation during the yeast cell cycle. *Science*, 307(5710):724–727, Feb 2005.
4. M Farkasovsky and H Kuntzel. Yeast Num1p associates with the mother cell cortex during S/G2 phase and affects microtubular functions. *J Cell Biol*, 131(4):1003–1014, Nov 1995.
5. M Farkasovsky and H Kuntzel. Cortical Num1p interacts with the dynein intermediate chain Pac11p and cytoplasmic microtubules in budding yeast. *J Cell Biol*, 152(2):251–262, Jan 2001.
6. D A Fell and A Wagner. The small world of metabolism. *Nat Biotechnol*, 18(11):1121–1122, Nov 2000.
7. Nabil Guelzim, Samuele Bottani, Paul Bourgine, and Francois Kepes. Topological and causal structure of the yeast transcriptional regulatory network. *Nat Genet*, 31(1):60–63, May 2002.
8. Ulrich Guldener, Martin Munsterkotter, Matthias Oesterheld, Philipp Pagel, Andreas Ruepp, Hans-Werner Mewes, and Volker Stumpflen. MPact: the MIPS protein interaction resource on yeast. *Nucleic Acids Res*, 34(Database issue):436–441, Jan 2006.
9. R M Halpern, S Q Chaney, B C Halpern, and R A Smith. Nicotinamide: a natural inhibitor of tRNA methylase. *Biochem Biophys Res Commun*, 42(4):602–607, Feb 1971.
10. Kristy M Hawkins and Christina D Smolke. The regulatory roles of the galactose permease and kinase in the induction response of the GAL network in Saccharomyces cerevisiae. *J Biol Chem*, 281(19):13485–13492, May 2006.
11. Jun Huan, Wei Wang, Deepak Bandyopadhyay, Jack Snoeyink, Jan Prins, and Alexander Tropsha. Mining protein family-specific residue packing patterns from protein structure graphs. In *Eighth International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 308–315, 2004.
12. T Ideker, V Thorsson, J A Ranish, R Christmas, J Buhler, J K Eng, R Bumgarner, D R Goodlett, R Aebersold, and L Hood. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 292(5518):929–934, May 2001.
13. Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 313–320, 2001.
14. Ker-Chau Li. Genome-wide coexpression dynamics: theory and application. *Proc Natl Acad Sci U S A*, 99(26):16875–16880, Dec 2002.
15. Nicholas M Luscombe, M Madan Babu, Haiyuan Yu, Michael Snyder, Sarah A Teichmann, and Mark Gerstein. Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature*, 431(7006):308–312, Sep 2004.
16. A Nabetani, T Koujin, C Tsutsumi, T Haraguchi, and Y Hiraoka. A conserved protein, Nuf2, is implicated in connecting the centromere to the spindle during chromosome segregation: a link between the kinetochore function and the spindle checkpoint. *Chromosoma*, 110(5):322–334, Sep 2001.
17. M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167, 2003.
18. Christos A. Ouzounis and Peter D. Karp. Global Properties of the Metabolic Map of Escherichia coli. *Genome Res.*, 10(4):568–576, 2000.
19. Susanne Prinz, Iliana Avila-Campillo, Christine Aldridge, Ajitha Srinivasan, Krassen Dimitrov, Andrew F Siegel, and Timothy Galitski. Control of yeast filamentous-form growth by modules in an integrated molecular network. *Genome Res*, 14(3):380–390, Mar 2004. Comparative Study.
20. Suresh K Purushothaman, Janusz M Bujnicki, Henri Grosjean, and Bruno Lapeyre. Trm11p and Trm112p are both required for the formation of 2-methylguanosine at position 10 in yeast tRNA. *Mol Cell Biol*, 25(11):4359–4370, Jun 2005.
21. Alexander W. Rives and Timothy Galitski. Modular organization of cellular networks. *Proceedings of the National Academy of Sciences*, 100(3):1128–1133, 2003.
22. Quansong Ruan, Debojyoti Dutta, Michael S Schwalbach, Joshua A Steele, Jed A Fuhrman, and Fengzhu Sun. Local similarity analysis reveals unique associations among marine bacterioplankton species and environmental factors. *Bioinformatics*, 22(20):2532–2538, Oct 2006.
23. Victor Spirin and Leonid A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123–12128, 2003.
24. Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. BioGRID: a general repository for interaction datasets. *Nucleic Acids Res*, 34(Database issue):535–539, Jan 2006.
25. D. Thieffry, A. Huerta, E. Perez-Rueda, and J. Collado-Vides. From specific gene regulation to genomic networks: A global analysis of transcriptional regulation in escherichia coli, 1998.
26. Jessica D Tytell and Peter K Sorger. Analysis of kinesin motor function at budding yeast kinetochores. *J Cell Biol*, 172(6):861–874, Mar 2006.
27. Christian von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417(6887):399–403, May 2002. Comparative Study.

28. Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 721, 2002.
29. Xifeng Yan, X. Jasmine Zhou, and Jiawei Han. Mining closed relational graphs with connectivity constraints. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 324–333, 2005.