# Learning Patterns in the Dynamics of Biological Networks

Chang hun You, Lawrence B. Holder, Diane J. Cook
School of Electrical Engineering & Computer Science
Washington State University
Box 642752, Pullman, WA 99164-2752
{changhun, holder, cook}@eecs.wsu.edu

## ABSTRACT

Our dynamic graph-based relational mining approach has been developed to learn structural patterns in biological networks as they change over time. The analysis of dynamic networks is important not only to understand life at the system-level, but also to discover novel patterns in other structural data. Most current graph-based data mining approaches overlook dynamic features of biological networks, because they are focused on only static graphs. Our approach analyzes a sequence of graphs and discovers rules that capture the changes that occur between pairs of graphs in the sequence. These rules represent the graph rewrite rules that the first graph must go through to be isomorphic to the second graph. Then, our approach feeds the graph rewrite rules into a machine learning system that learns general transformation rules describing the types of changes that occur for a class of dynamic biological networks. The discovered graph-rewriting rules show how biological networks change over time, and the transformation rules show the repeated patterns in the structural changes. In this paper, we apply our approach to biological networks to evaluate our approach and to understand how the biosystems change over time. We evaluate our results using coverage and prediction metrics, and compare to biological literature.

## Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; J.3 [**Life and Medical Science**]: Biology and genetics

## General Terms

Algorithms

## Keywords

Dynamic Network Analysis, Graph Mining, Biological Network, Graph Rewriting Rule

*KDD'09,* June 28–July 1, 2009, Paris, France.
Copyright 2009 ACM 978-1-60558-495-9/09/06 ...$5.00.

## 1. INTRODUCTION

There are many data that can be represented as graphs, where vertices represent entities and edges represent relationships between entities. Moreover, many of them have dynamic properties such that the structure of graphs can be changed over time. Our bodies are well-organized and vigorous systems, which promote reproduction and sustain our lives. These well-organized systems can be defined by the attributes and structural properties of biological networks, which include various molecules and relationships between molecules. Vigorous systems refer to dynamic properties of biological networks, which continuously change, while an organism performs various biological activities. Therefore, analysis of the dynamics of biological networks is necessary to understand biosystems.

Our approach first learns how one graph is structurally transformed into another using graph rewriting rules, and abstracts these rules into abstract patterns that represent the dynamics of a sequence of graphs. Our goal is to describe how the graphs change over time, not merely whether they change or by how much. In this way, our approach can help us understand the dynamics of biological networks.

This paper introduces our definition of graph-rewriting rules and more general transformation rules. We also present our two step algorithm to discover graph-rewriting rules in a dynamic graph, and transformation rules in the discovered graph-rewriting rules. In our experiments, we generate several dynamic graphs using the KEGG pathway database [9] in combination with the artificial generation and real data sets. We apply our approach to the pathways to understand how the biosystems change over time. We evaluate our results using coverage and prediction metrics, and compare to biological literature. Our results show important patterns in the dynamics of biological networks, i.e., discovering known patterns in the networks. Results also show the learned rules accurately predict future changes in the networks.

## 2. RELATED WORK

A graph is a natural way to represent biological networks. There are several graph mining approaches to biological networks [10, 11, 24]. These approaches represent biological networks as graphs, where vertices represent molecules and edges represent relations between molecules, and discover frequent patterns in these graphs. They discover structural features of networks, but they overlook temporal properties.

There is much research work on the dynamics of biosystems, such as mathematical modeling [16] and microarray analysis [22]. Mathematical modeling is an abstract model

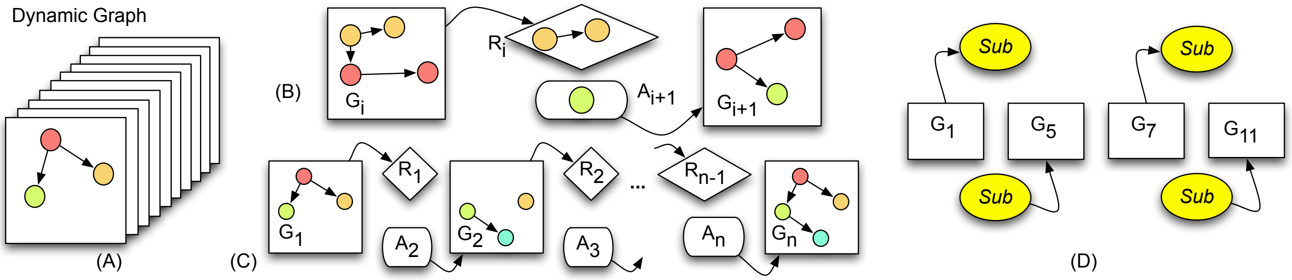

**Figure 1: A framework of dynamic graph analysis. (A) A dynamic graph (B) Learning graph rewriting rules from two sequential graphs. (C) Learning the entire set of graph rewriting rules. (D) Learning a transformation rule to abstract the learned graph rewriting rules (e.g., $Sub$ is removed from $G_i$ and then added back in $G_{i+4}$).**

to describe a system using mathematical formulae. They model the kinetics of pathways and analyze the trends in the amounts of molecules and the flux of biochemical reactions. The microarray is a tool for measuring gene expression levels for thousands of genes at the same time [3, 15]. Microarrays can also monitor patterns in gene expression levels over a period of time or for the different conditions. Patterns in gene expression levels can represent changes in the biological status or distinguish two different states, such as the normal and disease state. However, these two approaches disregard the structural aspect of networks.

Temporal data mining attempts to learn temporal patterns in sequential data, which is ordered with respect to some index like time stamps [17]. Temporal data mining is focused on discovery of relational aspects in data such as discovery of temporal relations or cause-effect association so that we can understand how or why the object changes rather than merely static properties of the object. Temporal data mining approaches discover temporal patterns in data, but they disregard relational aspects among entities.

Several methods have addressed dynamic graph analysis. Sun et al. [19] propose a technique to discover communities and detect changes in dynamic graphs that is represented as matrix and encoding schemes. Tensor analysis is also applied to dynamic graphs [20, 21]. Other work [1, 2, 18] proposes several detection measures of abnormal changes in the sequence of graphs and graph distance measures between two graphs. They can measure how much two graphs are different, but not show how they are different. Lahiri et. al. [13, 14] introduce an approach to predict the future structure in a dynamic network and mine periodic patterns using frequent subgraphs. Our approach uses a compression-based metric instead of the frequency-based approach to discover patterns in a dynamic graph.

## 3. PROBLEM DEFINITIONS

In this section, we define the graph rewriting rule and the transformation rule to describe the dynamic of a graph. Graph rewriting rules represent topological changes between two sequential versions of the graph, and transformation rules abstract the graph rewriting rules into the repeated patterns that represent the dynamics of the graph. Figure 1 shows a framework of our approach. The dynamic graph contains a sequence of graphs that are generated from sampling snapshots of the graph from a continuously-changing graph, i.e., a sequence of graphs represent one biological network that changes its structure over time. First, our approach learns graph rewriting rules including removals ($R_i$) and additions ($A_i$) between two sequential graphs $G_i$ and $G_{i+1}$ (figure 1 (B)), and generates a list of the entire graph rewriting rules (figure 1 (C)). Then, the final step is to learn the transformation rules to abstract the structural change of the dynamic graph based on the repeated patterns in the graph rewriting rules.

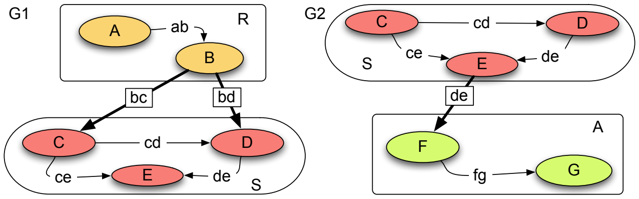

**Figure 2: An instance of graph rewriting rules between graph G1 and G2.**

### 3.1 Graph Rewriting Rules

First, we briefly describe graph rewriting rules for our approach with an example in figure 2. In our research, a graph $G$ denotes the directed labeled graph that is defined as $G = (V, E, L_v(V), L_e(E))$, where $V$ is a set of vertices, $E$ is a set of edges. To discover graph rewriting rules between two graphs, we first discover maximum common subgraphs (denoted by $S$) between two sequential graphs $G_1$ and $G_2$. Then, we derive removal (remainder in $G_1$ denoted by $R$) and addition subgraphs (remainder in $G_2$ denoted by $A$). Our graph-rewriting rules also contain connection edges. The connection edges are edges, which are used to link removal (or addition) subgraphs to the original graphs. The edges with boxed labels in figure 2 represent the connection edges between $G_1$ ($G_2$) and removal subgraph $R$ (addition subgraph $A$). The connection edges are important because they show how the subgraphs are connected to the original graphs. There can be more than one connection edge linking one subgraph to the original graph. The connection edges represent relations between the learned patterns and other elements in the input networks.

Formally, we define $DG = \{G_1, G_2, \cdots, G_n\}$ as a dynamic graph, where each graph $G_i$ is a graph at time $i$ for $1 \leq i \leq n$. For two consecutive graphs $G_i$ and $G_{i+1}$, we define $S_{i,i+1}$ as the maximum common subgraph between $G_i$ and $G_{i+1}$. $S_{i,i+1}$ can be a disconnected graph, i.e., describing the set of connected subgraphs common to $G_i$ and $G_{i+1}$. Then, we define a graph rewriting rule $GR_{i,i+1}$ as follows.

$$GR_{i,i+1} = \{(R_i, C_{R_i}), (A_{i+1}, C_{A_{i+1}})\}$$

Then, a removal subgraph $R_i$ and an addition subgraph $A_{i+1}$ are defined as follows.

$$R_i = G_i \backslash S_{i,i+1}, \; A_{i+1} = G_{i+1} \backslash S_{i,i+1}$$

$C_{R_i}$ and $C_{A_{i+1}}$ are the sets of connection edges for $R_i$ and $A_{i+1}$ respectively. The graph rewriting rule $GR_{1,2}$ in figure 2 can be represented as follows.

$$\begin{aligned} GR_{1,2} \;\; = \;\; & \{(R_1, \{(s2, g3, bc), (s2, g4, bd)\}), \\ & (A_2, \{(g3, s1, de)\})\}, \end{aligned}$$

The graph $R_1$ denotes $R$ (in $G_1$) that is linked by two connection edges labeled by 'bc' and 'bd'. $A_2$ denotes $A$ (in $G_2$) that is linked by one connection edge labeled by 'de'. In each edge, $sX$ and $gY$ denote the starting and ending vertices, where $s$ denotes the vertex in the subgraph and $g$ denotes the vertex in the original graph.

After iterating this process for $n$ graph, i.e., the entire sequence in the dynamic graph, we have $n-1$ $R$s and $n-1$ $A$s as shown in figure 1 (C). Here, we consider a set of graphs $L$ that is a list of graph rewriting rules learned in $DG$. $L$ contains $n-1$ $R$s and $n-1$ $A$s like $L = \{R_1, A_2, R_2, A_3, \cdots, R_{n-1}, A_n\}$. We arrange $R$ and $A$ in order of time when the event occurs.

## 3.2 Transformation Rules

Next, we discover transformation rules in the learned graph rewriting rules to abstract the structural changes in the dynamic graph as shown in figure 1 (D). A *transformation rule* is defined as a pattern in the learned graph rewriting rules, where the pattern best abstracts (compresses) the learned graph rewriting rules to best describe structural changes. More description will be in section 4. If some structural changes are repeated in the dynamic graph, there exist common subgraphs in the $R$s and $A$s. Then, we can discover the common patterns over $L$ as our transformation rules. Biologically speaking, if there exists a repeated change of the structure of a biological network, the change can be an important pattern in the network. Here, we propose one simple transformation rule $TR$, which represents repeated additions and removals (or vice versa), as follows.

$$TR_e = Sub_e\langle +t_a, -t_r \rangle$$

In the case when the transformation rule represents only repeated removals (or additions), $-t_r$ (or $+t_a$) would be $\emptyset$, like $Sub\langle -t_r \rangle$ (or $Sub\langle +t_a \rangle$). $Sub$ represents a subgraph, which adds to and/or removes from the graph repeatedly. $+t_a$ represents the time interval from the last removal to the current addition, and $-t_r$ represents the time interval from the last addition to the current removal. If $+t_a$ is shown before $-t_r$, the addition precedes the removal. For instance, $Sub\langle +4, -2 \rangle$ denotes a repeated structure added after 4 time intervals from the last removal and removed after 2 time intervals from the last addition as shown in figure 1 (D). $e$ denotes the number of the transformation rules in one dynamic graph. There can be multiple patterns over $L$ to describe the structural change of the dynamic graph, where the best transformation rule that is labeled as $TR_1$ best describes the change.

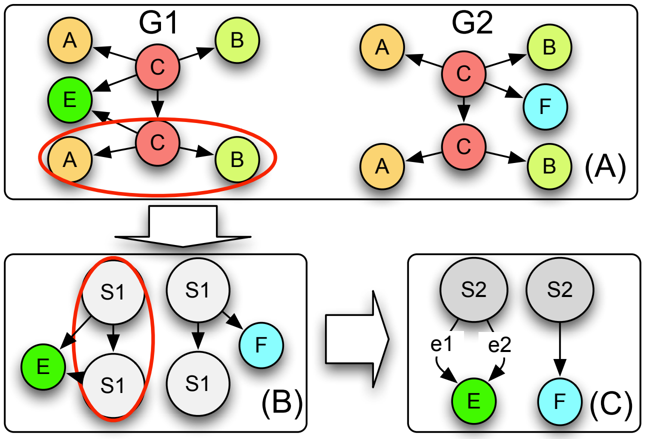

**Figure 3: Discovery of the best compressed subgraph in a set of graphs at iteration 1 (A), 2 (B), and 3 (C).**

There are other forms of transformation rules besides repeated add/remove rules, such as patterns conditional on context, i.e., removal/addition of structure $X$ if structure $Y$ is present (or absent), or patterns that describe numeric changes in combination with structure, i.e., describing trends of concentration, not just appearance. We will consider other types of transformation rules in future work.

## 4. APPROACH

This section describes our approach to analyze dynamic graphs. We present a two step algorithm: Learning Graph Rewriting Rules and Learning Transformation Rules. Algorithm 1 learns graph rewriting rules in a dynamic graph to represent how two sequential graphs are different. Algorithm 2 learns the repeated transformation rules in the learned graph rewriting rules to describe how the graph changes over time, where the changes are actually represented as a sequence of revised graphs. For both algorithms we rely on a previously-developed method for finding the best-compressing subgraph in a set of graphs. For the first algorithm, repeated application of this method allows us to find the set of all subgraphs common to a pair of consecutive graphs. For the second algorithm this method allows us to find the subgraphs repeatedly added and removed in the dynamic graph. While we could use a frequent subgraph miner [12, 23] for this purpose, experiments have shown that the best-compressing patterns comparably capture the complete repeated structural changes [25].

We define the best-compressing subgraphs as those which minimize the description length of the input graph after being compressed by the subgraphs based on the Minimum Description Length (MDL) principle [4, 5]. Formally, the description length of the substructure $S$ is represented by $DL(S)$, the description length of the input graph is $DL(G)$, and the description length of the input graph after compression is $DL(G|S)$. The approach finds a substructure $S$ that minimizes the $Compression$ of the graph defined as follows.

$$Compression = \frac{DL(S) + DL(G|S)}{DL(G)}$$

Figure 3 shows an example of the subgraph discovery by this compression-based approach. First, we can discover four instances of one common subgraph denoted by a red circle (A). After discovery, we compress each instance replacing by one vertex ($S_1$), and we iterate the discovery process. In the second iteration (B), we discover two instances of the next common subgraph, and compress them by one vertex ($S_2$). We stop the iteration because there is no more common subgraph, i.e., no more compression (C).

## 4.1 Learning Graph Rewriting Rules

```
Algorithm 1 : Learning Graph Rewriting Rules
Input: Dynamic graph $DG = \{G_1, G_2, \cdots, G_n\}$
Output: Rewrite rules $L$, connection edges$C$
 1: $L = \{\}, C = \{\}$
 2: for $i = 1$ to $n - 1$ do
 3:   $Graphs = \{G_i, G_{i+1}\}$, $S = \{\}$
 4:   while More compression possible do
 5:     $BestSub = DiscoverCommonSub$ in $Graphs$
 6:     $S = S \cup BestSub$
 7:     Compress $Graphs$ by $BestSub$
 8:   end while
 9:   Find $R_i = G_i \backslash S$ and $C_{R_i}$ in $G_i$
10:   Add $R_i$ into $L$, and Add $C_{R_i}$ into $C$
11:   Find $A_{i+1} = G_{i+1} \backslash S$ and $C_{A_{i+1}}$ in $G_{i+1}$
12:   Add $A_{i+1}$ into $L$, and add $C_{A_{i+1}}$ into $C$
13: end for
```

Using the compression-based approach (as *DiscoverCommonSub* in the algorithm), we describe our two step algorithm. Algorithm 1 shows the learning graph rewriting rules algorithm, where the entire algorithm denotes figure 1 (C) and the each iteration in the outer loop denotes figure 1 (B). First, the algorithm initialize $L$ and $C$ to store removal and addition subgraphs, and connection edges. At line 3, the algorithm prepares two sequential graphs as *Graphs*, and then discovers one common subgraph by the compression-based approach. After compression, the algorithm discovers another subgraph at the next iteration until there is no more compression. In this way, the algorithm can discover the maximum common subgraph between two sequential graphs. After compressing the two graphs by the maximum common subgraph, the algorithm identifies removal (or addition) subgraphs and connection edges (lines 9 and 11) using a modified Breadth First Search (mBFS), which adds each edge as well as each vertex into the queues as visited or to be visited. After compression, each maximum common subgraph is replaced by one vertex $S_i$. mBFS starts to search from one edge linked to $S_i$ to find one disconnected subgraph, and the starting edge is added into $C$. During the search, if there is one more edge between the disconnected subgraph and maximum common subgraph, the edge becomes the other connection edge. In this way, mBFS can find all disconnected subgraphs (without considering the link by the connection edges), and they become removal (or addition) subgraphs. mBFS stops the search when all connected edges are added in $C$. For example, in figure 3 (C), mBFS starts from one edge linked to $S_2$ (in case of $G_1$, choose $e_1$), and these starting edges are added into in $C$. Since there is one more linked edge ($e_2$) to $S_2$ in case of $G_1$, $e_2$ is added into $C$. Then, there is no place to visit from the vertex $E$, $E$ becomes a disconnected subgraph as an addition subgraph. Since there is no place to visit from the vertex $F$ in $G_2$, $F$ becomes a disconnected subgraph as a removal subgraph. In this way, mBFS identifies removal subgraphs $R_i$ and addition subgraphs $A_{i+1}$ with connection edges. The output of Algorithm 1 includes $L$ and $C$. $L$ and $C$ are bijective. $L = \{R_1, A_2, \cdots, R_{n-1}, A_n\}$ is used in Algorithm 2 as an input. $C = \{C_{R_1}, C_{A_2}, \cdots C_{R_{n-1}}, C_{A_n}\}$ is used to visualize the relations between the learned subgraphs and original graphs.

## 4.2 Learning Transformation Rules

```
Algorithm 2 : Learning Transformation Rules
Input: $L$, $Iter$
Output: $BestCommonSubs$,$ListOfDist$
1: while More compression possible and $Iter > 0$ do
2:   $BestSub = DiscoverCommonSub$ in $L$
3:   Add $BestSub$ into $BestCommonSubs$
4:   Calculate $distance$ between instances of $BestSub$
5:   Add $distance$ into $ListOfDist$
6:   Compress $L$ by $BestSub$
7:   $Iter = Iter$ -1
8: end while
```

From the result of Algorithm 1, we try to discover repeated rewrites as our transformation rules to better understand how graphs change over time as shown in figure 1 (D). The input $L$ contains $2(n-1)$ graphs: $n-1$ $R$s and $n-1$ $A$s. Note that each example (each $R$ or $A$) contains one or more graphs, which may not be connected to each other. We then use *DiscoverCommonSub* again to find common subgraphs in $L$ (line 2). As described in figure 3, the best common subgraph in $L$ represents the subgraph in our transformation rule. We calculate the temporal *distance* between two consecutive instances of the best-compressing subgraphs to describe the time at which the removal (or addition) occurs after the previous addition (or removal) at line 4. After the discovery of the common subgraph, $L$ is compressed by this subgraph (line 6), and the discovery process is iterated until no more compression is achieved or we reach a user-defined limit $Iter$ on the number of iterations. When the best subgraph at a latter iteration includes the best subgraph from a former iteration, the results can show the latter best subgraph includes a previously-learned subgraph that is replaced by one vertex. More detail will be described with examples in the results section. In $TR_e$, the $e$ denotes the number of iterations. If a transformation rule is discovered in the first iteration, the rule is labeled as $TR_1$ that is the best subgraph in $L$. If $Iter$ is not specified, Algorithm 2 finds all possible $TR$ in $L$.

## 4.3 Complexity Issue

One challenge of our algorithm is to discover maximum common subgraphs between two sequential graphs, because this problem is known to be NP-complete [8]. To address this issue we use a parameter, *limit*, in *DiscoverCommonSub* to restrict the number of substructures to consider in each iteration. We can express the Algorithm 1's total runtime as $N_1 = N_{DCS}(T-1)$, where $N_{DCS}$ is the runtime of *DiscoverCommonSub* and it runs for T-1 times. Algorithm 2's running time is dominated by $N_{DCS}$. $N_{DCS}$ is restricted by *limit* that is calculated based on input data, specifically, the number of unique vertex and edge labels. A previous work [6] shows $N_{DCS}$ running with a fully-connected graph in time polynomial with *limit*. We can avoid the worst case in our domain, because biological networks are usually sparse graphs and there are not many instances due to plenty of unique labels. But we still need to pursue reducing the running time for other domains. Also, our algorithm does not try to discover the entire set of maximum common substructures at once. In each step, the algorithm discovers a common, connected substructure and iterates the discovery process until discovering the entire set.

Graphs that represent biological networks usually contain unique vertex labels, because each vertex label usually denotes the name of the molecule. Because the maximum common subgraph problem in graphs with unique vertex labels is known to have quadratic complexity [7], discovery of the graph rewriting rules is still feasible. However, there will be a tradeoff between exactness and computation time when analyzing very large graphs.

## 4.4 Evaluation Metrics

We use two metrics to evaluate the learned transformation rules. The first metric is $Coverage$ that represents how well the rule describes the changes in the graphs. The $Coverage$ of the $BestSub$ discovered at iteration $i$ in Algorithm 2 is computed as follows.

$$Coverage = \frac{size(BestSub) \sum_{g \in coveredAs,Rs} \frac{1}{size(g)}}{2(n-1)}$$

where the covered $A$s and $R$s are the addition and removal subgraphs in $L$ that contain $BestSub$. The size of a graph $G$ is calculated as $size(G) = |V| + |E|$. These graphs are efficiently identified during the discovery of $BestSub$, avoiding the need for costly subgraph isomorphism tests. $Coverage$ represents the portion of the learned subgraphs (the removal or addition subgraphs) described by the transformation rule to be based on $BestSub$. For example, suppose we have $n = 3$ graphs from which we find two graph-rewriting rules. Then, we have two removal and two addition subgraphs. Assume the size of $R_1$ is 10, $R_2$ is 12, $A_2$ is 10, and $A_3$ is 15. Also assume the $BestSub$ is found in $R_1$ and $A_2$, the $BestSub$ has a size of 5. $Coverage$ is computed as $\frac{5(1/10+1/10)}{4} = 0.25$. Higher $Coverage$ indicates the subgraph can describe more significant (larger portions of) changes. Currently, $Coverage$ does not consider the size of connection edges ($|C|$). Unless the subgraph is isomorphic with all AGs and RGs, $Coverage < 1$.

We define $Prediction$ as our second metric to evaluate the prediction capability of the learned transformation rules as follows.

$$Prediction = \frac{\sum_{i \in P} d(RealSub_i, PredictedSub_i)}{|P|}$$

$P$ is the set of positions where we predict the $PredictedSub_i$ will show up, $RealSub_i$ is the actual subgraph found at position i, and $d(G_m, G_n)$ is defined as follows.

$$d(G_m, G_n) = \frac{|mcs(G_m, G_n)|}{|G_m \cup G_n|}$$

$d(G_m, G_n)$ is a graph distance metric by Bunke et. al. [2, 18], where $mcs(G_m, G_n)$ denotes the maximum common subgraph between $G_m$ and $G_n$. In contrast to their work that defines the size of $G$ as the number of vertices in $G$, we consider the number of vertices and edges defined in the previous paragraph. If two graphs $G_m$ and $G_n$ are isomorphic, $d(G_m, G_n) = 1$. For example, $d(G_1, G_2)$ in figure 3 is 11/16, where $mcs(G_1, G_2) = 11$ and $|G_m \cup G_n| = 16$. $Prediction$ represents how much the predicted subgraph covers the subgraphs in the testing experiments. For example, suppose we predict a subgraph $s$ will be shown 3 times in the testing data. Then, we discover the subgraph $rs$ that is partially different from $s$ at one time point ($(G_{rs}, G_s) = 0.5$), and isomorphic subgraphs with $s$ at another time point. $Prediction$ is computed as $\frac{0.5+1.0+0}{3} = 0.5$. Currently, our $Prediction$ measure is not for a temporal prediction, i.e., the exact time the subgraph appears, but for a sequential prediction, i.e., whether the correct sequence of the subgraphs appears.

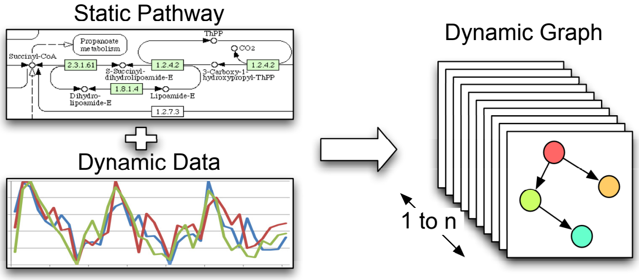

**Figure 4: The generation of a dynamic graph in combination with the data of the dynamic properties. If the data of the dynamic properties has $n$ time slices, the dynamic graph has $n$ graphs.**

# 5. EXPERIMENTS AND RESULTS

We perform four experiments to evaluate our approach using three ways: artificial generation, and combinations with two real world data sets. We generate a static graph representing the biological networks from the KEGG PATHWAY data [9], where vertices represent compounds, genes, enzymes, relations and reactions, and edges represent relationships between vertices. Then, we use our data sets to transform the static graph to a dynamic graph as shown in figure 4. In the artificial generation, we use a real biological network, but we remove and add some subgraphs manually to generate the dynamic graphs. In the real world data, we use the KEGG data [9] in combination with additional data to generate dynamic graphs. Because the KEGG data contains only the static structure of pathways, we need to use additional data including dynamic properties of pathways. We refer to results of two researches: one for the cell cycle signaling pathway with mathematical modeling [16] and the other for metabolic pathways with microarray data [22].

## 5.1 Artificial Generation

The biological network used in the artificial generations is the Notch signaling pathway in humans generated from the KEGG data. The Notch signaling pathway contains 46 genes in our experiments, and we assume that each gene can be shown at most once at each time slice. First, we create one list that contains the names of 46 genes, and then duplicate the list for 20 time slices. For varying several conditions, we remove one or more genes at specific times. Because of the biological semantics, the removal of even one gene can cause the removal of one or more larger subgraphs. We generate four dynamic graphs, each of which has 20 time slices. The size of each dynamic graph varies: 3,380 (164 to

**Table 1: *Coverage* of the best subgraphs in Artificial Data. Data denotes the artificial biological networks. The number in each iteration denotes $x$ ($y$), where $x$ denotes the number of the discovered subgraphs and $y$ denotes the *Coverage* by the best subgraph discovered at the iteration. Total denotes the total *Coverage*.**

| Data | $TR_1$ | $TR_2$ | $TR_3$ | Total |
|---|---|---|---|---|
| $N_A$ | 19 (1.0) | NA | NA | 1.0 |
| $N_B$ | 9 (1.0) | NA | NA | 1.0 |
| $N_C$ | 8 (0.16) | 4 (0.032) | 10 (0.05) | 0.242 |
| $N_D$ | 6 (0.15) | 5 (0.125) | 2 (0.045) | 0.320 |

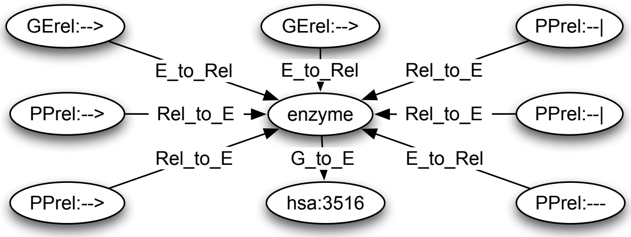

**Figure 5: The best subgraph discovered in the graph rewriting rules of the dynamic graph $N_B$.**

177) for $N_A$, 3,350 (149 to 174) for $N_B$, 2,733 (102 to 174) for $N_C$ and 3,332 (152 to 174) for $N_D$. The numbers in () denote the minimum size and maximum size of a graph in a dynamic graph respectively.

The goal of the artificial generation experiment is to identify the strengths and weaknesses of our approach. Table 1 shows the coverage of the best subgraph (our rule) discovered at each iteration of Algorithm 2. The first two dynamic graphs, $N_A$ and $N_B$, can be represented by one transformation rule, because the removals and additions are simple and regular. Generally, the structural change in the dynamic graph is represented by multiple transformation rules like $N_C$ and $N_D$. For example, $N_C$ is represented by $TR_1$ as a portion of the coverage 0.16. But $N_A$ is fully covered by $TR_1$, i.e., $TR_1$ can describe the whole structural change.

Figure 5 shows the best subgraph discovered in the $N_B$ experiments. The instances of the best subgraphs are discovered in the 9 examples (4 removals and 5 additions). "GErel" denotes the relation between a gene and protein, and "PPrel" denotes the relation between two proteins. Therefore, the enzyme generated by a gene, hsa:3516, has 7 relations, such as 2 relations to other genes and 5 relations to other proteins. The transformation rule including this subgraph can be visualized as shown in figure 6. The above rhombuses denote the removals at the specified time. The below eclipses denote the additions at the specified time. The numbers on the arrow denote the temporal distance between two events: removals and additions. The first addition occurs at time 1, and the first removal occurs after 3 time intervals. From the first addition at time 1 to the last addition at time 17, every removal is repeated after 3 time intervals from the last addition, and every addition is repeated after 1 time interval from the last removal. The repeated transformation rule can be represented as shown in figure 6 and can be expressed as $TR_1 = Sub_1\langle +3, -1\rangle$.

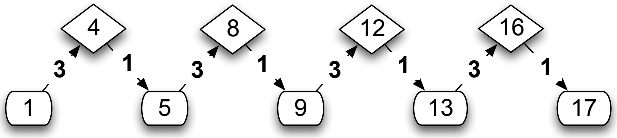

**Figure 6: Visualization of transformation rules including the subgraph in figure 5.**

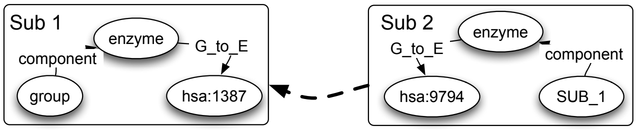

**Figure 7: Two best subgraphs discovered in $N_D$. $Sub_1$ is discovered at times 3, 5, 8, 10, 15, and 18, and $Sub_2$ is discovered at times 3, 5, 8, 15 and 18. $Sub_1$ is included into $Sub_2$ as a previously-learned subgraph.**

As described in section 4.2, figure 7 shows an example of a previously-learned subgraph that $Sub_2$ includes $Sub_1$ discovered in $N_D$. At the first iteration (as $TR_1$), the first subgraph ($Sub_1$) is discovered at times 5, 10, 15 as removals and at times 3, 8, 18 as additions. Then, this subgraph is compressed and replaced by one vertex labeled by "Sub_1". At the second iteration (as $TR_2$), the second subgraph ($Sub_2$) is discovered at times 5, 15 as removals, and at times 3, 8, 18 as additions. Because $Sub_2$ includes $Sub_1$, $Sub_1$ is included into $Sub_2$ as a vertex "Sub_1". In figure 7, the dashed-line arrow represents a pointer to the previously-learned subgraph $Sub_1$ from $Sub_2$. Biologically hsa:1387 in $Sub_1$ and hsa:9794 in $Sub_2$ are included into a "group" ($Sub_1$) as "component"s.

Here, we discuss the advantage of the compression-based subgraph discovery. In $N_C$, the first best subgraphs are discovered 8 times. Actually, the third best subgraphs are discovered 10 times. Because the *Compression* of the first subgraph is better than the *Compression* of the third subgraph, our approach prefers the first subgraph. A frequency-based approach would prefer the third subgraph. The size of the first subgraph is 51, and the size of the third subgraph is 5. Also, the *Coverage* (0.16) of the first subgraph is larger than the *Coverage* (0.05) of the third subgraph. For this reason, the compression-based approach can be more useful than frequent graph mining in the analysis of dynamic graphs. The detailed comparison results are in [25].

## 5.2 Mathematical Modeling

We also apply our approach to a dynamic graph based on the mathematical modeling data. The dynamic graph represents the cell cycle signaling pathway [16]. The cell cycle signaling network in our experiment contains 14 molecules (genes and compounds) and 11 reactions between molecules. We use a threshold $th$ to activate each compound or gene. At each time, a compound or gene, which has more than $th$ amount, is shown in the graph. In other words, the biological network contains a portion of the 14 molecules with related reactions at each time. We normalize the concentrations of 14 molecules from 0 to 1, because we are focused on trends in the changes, and the concentrations of different molecules vary significantly. Because the simulation is performed for 700 seconds and we take a snapshot at every 10 seconds, we

**Table 2: Results of the prediction experiments with the modified model. Name denotes the name of the case. Variable denotes the name of the modified parameter. Mod. denotes the modification (X/Y), where X denotes the new value and Y denotes the default value. Size denotes the size of each dynamic graph. Transformation Rule denotes the learned transformation rule. Sub1 size denotes the size of the subgraph in the transformation rule. Coverage denotes the *Coverage* of the learned rule, and Prediction represents the *Prediction* of the learned rule.**

| Name | Variable | Mod. | Size | Transformation Rule | Sub1 size | Coverage | Prediction |
|---|---|---|---|---|---|---|---|
| $M_1$ | k1 | 200/300 | 645 | $Sub_1\langle +8, -0\rangle$ | 27 | 0.115 | 1.0 |
| $M_2$ | k2 | 3/5 | 1541 | $Sub_1\langle +5, -1\rangle$ | 30 | 0.153 | 0.962 |
| $M_3$ | k4 | 50/30 | 835 | $Sub_1\langle +13, -0\rangle$ | 27 | 0.051 | 1.0 |
| $M_4$ | k5 | 0.2/0.1 | 1530 | $Sub_1\langle +5, -1\rangle$ | 25 | 0.155 | 1.0 |
| $M_5$ | k7 | 6/10 | 1880 | $Sub_1\langle +6, -4\rangle$ | 28 | 0.084 | 1.0 |
| $M_6$ | k8 | 60/100 | 1007 | $Sub_1\langle +9, -0\rangle$ | 27 | 0.080 | 0.864 |
| $M_7$ | k10 | 20/10 | 1741 | $Sub_1\langle +5, -2\rangle$ | 27 | 0.119 | 0.852 |
| $M_8$ | k11 | 0.5/1 | 1003 | $Sub_1\langle +10, -0\rangle$ | 27 | 0.066 | 0.944 |
| $M_9$ | k2u | 300/50 | 886 | $Sub_1\langle +19, -0\rangle$ | 27 | 0.033 | 1.0 |
| $M_{10}$ | tau | 15/25 | 1402 | $Sub_1\langle +4, -0\rangle$ | 27 | 0.185 | 1.0 |
| Average | _ | _ | _ | _ | _ | 0.1041 | 0.962 |

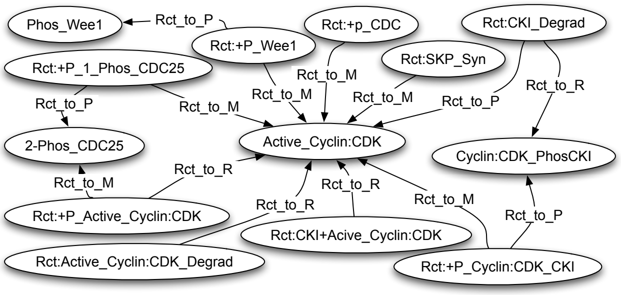

**Figure 8: The best subgraph ($Sub_1$) discovered in $TR_1$.**

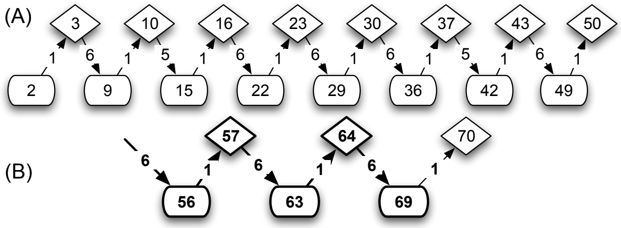

**Figure 9: Visualization of the graph rewriting rules including the subgraph in figure 8 in the training data (A) and the testing data (B).**

have 51 time slices ($t = 1$ to 51) of data for training and the following 20 time series for testing.

Figure 8 shows the best subgraph ($Sub_1$) in $TR_1$ discovered at 16 time slices as visualized in figure 9 (A). The vertices containing "Rct" in the labels denote reactions like "Rct:+p_CDC". The vertices without "Rct" denote molecules (genes or proteins). The three edges, "Rct_to_R", "Rct_to_P" and "Rct_to_M", denote how the molecules are related to the reactions as reactant, product and modifier respectively. These results are biologically significant, because they describe the repeated structural changes in the networks. Qu et al. [16] describe periodic changes of molecules (i.e., amount of molecules). Specifically, they mention several molecules such as Active Cyclin:CDK and Free Cyclin that show periodic increase and decrease, where the cycles correspond to the change of the cell size. Figure 9 (A) shows the subgraph including Active Cyclin:CDK, that is added and removed periodically corresponding to periodic changes in the amount of the molecule. In addition, figure 8 show how the changes are related to other elements (i.e., which elements are removed or added at the same time) as shown in the discovered subgraphs and how the subgraphs are linked to the original graphs. Our results show patterns in the structural changes, not merely changes of amount.

The *Coverage* is calculated as 0.181. Based on this rule, we predict the future change as shown in figure 9 (B). We predict 6 graph rewriting rules (future changes), as we choose the predicted temporal distance based on the distances observed in training. The temporal distance and graph rewriting rules denoted by the bold fonts represent the same patterns with the testing data. 5 patterns out of 6 predictions are same as training data. The only 6th pattern at time 70 is a non-isomorphic graph with $Sub_1$. $d(Sub_1, R_{70})$ is computed as 0.833, and the *Prediction* is 0.972.

## 5.3 Prediction Experiment

Next, we process a simple prediction experiment. Because our research is focused on patterns in graph rewriting rules (i.e., patterns in structural changes), we can predict which graph rewriting rules appear (i.e., which structural changes occur). To evaluate prediction ability of the learned transformation rules, we perform ten prediction experiments using the above modeling data.

We modify some initial parameters in the model to generate different dynamic graphs. The modified parameters and values are shown in table 2. Like the above modeling experiment, we use 51 time series as training and 20 time series as testing. Table 2 shows the results. $M_1$ shows the transformation rule $Sub_1\langle +8, -0\rangle$ that describes $Sub_1$ is added after 8 times from the last removal and is removed right after the last addition. For example, $Sub_1$ is added at time 12 (during the time from 11 to 12), and is removed at time 12 (during the time from 12 to 13).

As shown in table 2, the averages of the rule coverage and prediction coverage are larger than 0.9, indicating that our approach is able to learn accurate rules across the different conditions yielding different dynamic graphs. In case of

**Table 3: Dynamic graphs of metabolic pathways and results. Name denotes the KEGG IDs of pathways represented by the dynamic graphs. The second to fifth column show the information of pathways, such as the number of compounds (cpd), genes (gene), reactions (rct) and relations (rel). Max. denotes the maximum size of one graph in the dynamic graph. Min. denotes the minimum size of one graph in the dynamic graph. Total denotes the size of the dynamic graph. Rule denotes the subgraph is included in the transformation rule. Coverage denotes the *Coverage* of the transformation rule. Run denotes the running time (seconds).**

| Name | # cpd | # gene | # rct | # rel | Max. | Min. | Total | Rule | Coverage | Run (sec.) |
|---|---|---|---|---|---|---|---|---|---|---|
| 00020 | 20 | 30 | 17 | 73 | 251 | 46 | 3,483 | $Sub_3$ | 0.024 | 10.14 |
| 00230 | 73 | 172 | 70 | 161 | 618 | 134 | 7,861 | $Sub_2$ | 0.048 | 138.78 |
| 00330 | 19 | 14 | 21 | 25 | 176 | 60 | 3,528 | $Sub_1$ | 0.055 | 11.78 |
| 00564 | 23 | 23 | 21 | 38 | 203 | 56 | 3,695 | $Sub_4$ | 0.027 | 12.67 |

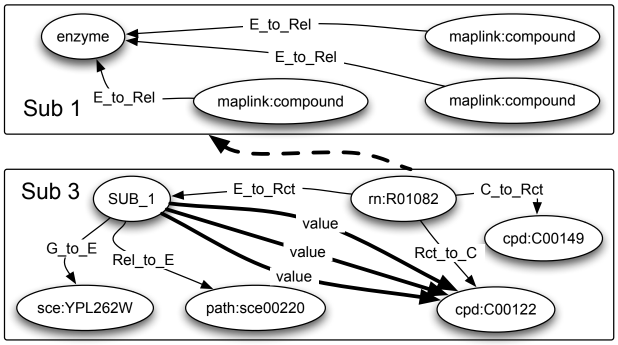

**Figure 10: Two best subgraphs discovered in the experiment of the TCA cycle with the microarray data. $Sub_1$ is included into $Sub_3$ as a previously-learned subgraph.**

the $M_1$, $M_3$, $M_5$, $M_6$ and $M_8$, they show relatively small coverage, because some elements in the best subgraph are removed (or added) separately. The detail discussion of this problem is in [25]. $M_9$ contains the entire sequence of discovered subgraphs in the transformation rule, but the oscillation in $M_9$ shows only two cycles. In most cases, the oscillation shows more than 5 cycles (i.e., figure 9). Our algorithm can predict the future structural changes from the learned transformation rules of the graph rewriting rules that represent the structural changes of dynamic graphs. We will compare our result with other approaches in the future work.

## 5.4 Microarray Data

Now, we show the result of the dynamic graphs based on microarray data. Table 3 shows brief information of the dynamic graphs and results. In previous results we show $TR_1$. Here, it is a little bit different, because the pathway is bigger than the previous cases and contains many redundant labels. In the aspect of the dynamic graph mining, $TR_1$ including $Sub_1$ best describe the structural change. Biologically, $TR_1$ is too general to describe the structural change. In figure 10, $Sub_1$ that is discovered 46 times at 20 time points contains only general information: three maplink-relations (relation between a gene (protein) and pathway) and one enzyme. Without any specific name of gene or pathway, $Sub_1$ represents too general information. For this reason, we show $Sub_3$ (as $TR_3$) in figure 10 that contains any specific name of the gene, because our microarray data represent the trends of the gene expression values, and the gene is the only information that can be changed over time. $Sub_1$ is included into $Sub_3$ as a previously-learned subgraph. $Sub_3$ includes one gene (YPL262W) and one pathway (sce00220) and one reaction (R01082) and two compounds (C00122 and C00149).

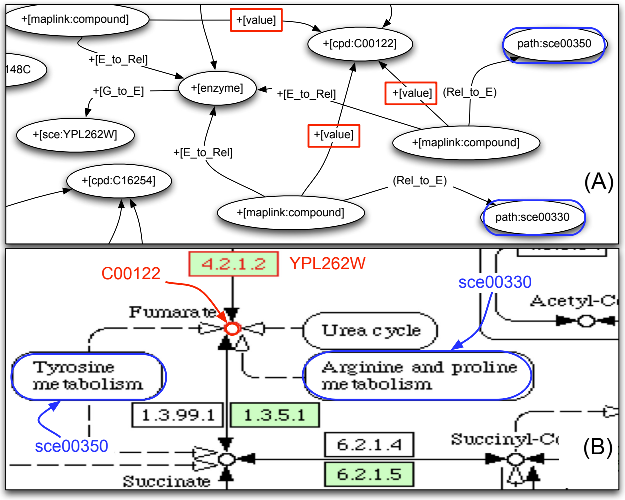

**Figure 11: (A) A visualization of $Sub_3$ in figure 10. (B) $Sub_3$ on a portion of the TCA cycle pathway map.**

$Sub_3$ is discovered as removals at time 14, 23, 26 and as additions at time 2, 23, 25. Because the original microarray research [22] has only 36 time series, we do not perform the prediction task. But this experiment shows that our approach can be applied to real data, because the microarray data is generated from the yeast cells. The original result of microarray shows more than 50% of genes have three periodic cycles in the gene expression. In our experiment, the appearance of most learned graph rewriting rules in four pathways also shows three periodic cycles like $Sub_3$.

Figure 11 shows the visualization of $Sub_3$ from figure 10 to describe biological meaning of structural patterns. (A) shows an addition rule in our output, and (B) shows the same rule marked on the KEGG pathway map [9]. The labels marked by "+[] (-[])" represent the labeled vertices and edges belonging to the subgraphs of addition rules (removal rules). Connection edges between the discovered substructures and original graphs are marked by "()". In figure 10, we can notice the three edges labeled by "value" linked to C00122, which are from the three "maplink" vertices in $Sub_1$. These three edges are marked by the red boxes in figure 11 (A). The maplink denotes a relation between one gene (X) in a pathway (Y) and another pathway (Z). The compound that is linked to "maplink"-relation by "value" edge denotes a compound shared in two pathways (Y and Z). Precisely, the compound has two relations with a gene (X) and another gene (that cannot be known at this point) in pathway (Z). Figure 11 (A) can help us understand these relationships. C00122 is added at time 25 with relations to three maplink-relations. Two relations out of three maplink-relations are connected to the other two pathways (sce00330 and sce00350) marked by the blue eclipses. These two pathways are not marked by "[]" or "()", because they already exist before time 25. In other words, $Sub_3$ is added at time 25, and connected to two pathways by two connection edges.

Microarray data [22] can show three periodic cycles in the change of the gene expression values. Our approach also can discover three periodic cycles of removals and additions of the genes (i.e., YPL262W). In addition to the three periodic cycles of removal and addition of one element, our results also show what other elements are related to the removed (or added) genes, i.e., how the removed (or added) genes relate to others in the pathway. The connection edge can help us understand how the learned subgraphs relate to the original graph at each time.

# 6. CONCLUSION

This research introduces the use of graph rewriting rules to describe structurally changing networks, and more general transformation rules abstracting the graph rewriting rules. We also present a two step algorithm to discover graph rewriting rules and transformation rules in a dynamic graph. The algorithm is evaluated with the dynamic graphs representing the biological networks in combination with the artificial generation, mathematical modeling and microarray data. The graph rewriting rules show how one graph is transformed into another. The learned transformation rules over the graph rewriting rules can describe repeated patterns in the series of the structural changes.

Our results show important patterns in the dynamics of biological networks, for example, discovering known patterns in the various networks. Results also show the learned rules accurately predict future changes in the networks. The connection edge can help us understand how the learned subgraphs relate to the original pathway at each time. Our approach also helps us visualize the change of subgraphs at each time to show how the networks structurally change, helps us better explore how networks change over time, and guides us to understand the structural behaviors of the dynamic network.

For our future work we will explore a better approach to learn transformation rules that can cover graph rewriting rules that are divided over several consecutive time slices. Also, our prediction measure needs to include a temporal distance factor to better evaluate rules in terms of predicting the precise time at which a change occurs.

# 7. REFERENCES

[1] H. Bunke, M. Kraetzl, P. Shoubridge, and W. Wallis. Detection of abnormal change in time series of graphs. *J. of Intercon. Net.*, 3, Nos 1+2:85–101, 2002.

[2] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Lett.*, 19(3-4):255–259, 1998.

[3] H. Causton, J. Quackenbush, and A. Brazma. *A Beginner's Guide Microarray Gene Expression Data Analysis*. Blackwell, 2003.

[4] D. Cook and L. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of AIR*, 1:231–255, 1994.

[5] D. Cook and L. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.

[6] D. Cook, L. Holder, and S. Djoko. Scalable discovery of informative structural concepts using domain knowledge. *IEEE Expert*, 11:59–68, 1996.

[7] P. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl. On graphs with unique node labels. In *IAPR-GBR*, 2003.

[8] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[9] KEGG. http://www.genome.jp.

[10] M. Koyuturk, A. Grama, and W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. In *ISMB*, 2004.

[11] J. Kukluk, C. You, L. Holder, and D. Cook. Learning node replacement graph grammars in metabolic pathways. In *BIOCOMP*, 2007.

[12] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, 2001.

[13] M. Lahiri and T. Berger-Wolf. Structure prediction in temporal networks using frequent subgraphs. In *CIDM*, 2007.

[14] M. Lahiri and T. Berger-Wolf. Mining periodic behavior in dynamic social networks. In *ICDM*, 2008.

[15] D. J. Lockhart and E. A. Winzeler. Genomics, gene expression & DNA arrays. *Nature*, 405:827– 836, 2000.

[16] Z. Qu, W. MacLellan, and J. Weiss. Dynamics of the cell cycle: checkpoints, sizers, and timers. *Biophys J*, 85(6):3600–11, Dec 2003.

[17] J. F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE TKDM*, 14(4):750–767, 2002.

[18] P. Shoubridge, M. Kraetzl, W. Wallis, and H. Bunke. Detection of abnormal change in a time series of graph. *J. of Intercon. Net.*, 3:85–101, 2002.

[19] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *SIGKDD*, 2007.

[20] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *SIGKDD*, 2006.

[21] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos. Colibri: Fast mining of large static and dynamic graphs. In *SIGKDD*, 2008.

[22] B. Tu, A. Kudlicki, M. Rowicka, and S. McKnight. Logic of the yeast metabolic cycle: Temporal compartmentalization of cellular processes. *Science*, 310, 2005.

[23] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *ICDM*, 2002.

[24] C. You, L. Holder, and D. Cook. Application of graph-based data mining to metabolic pathways. In *ICDM Workshop on DMB*, 2006.

[25] C. You, L. Holder, and D. Cook. Graph-based data mining in dynamic networks: Empirical comparison of compression-based and frequency-based subgraph mining. In *ICDM Workshop on ADN*, 2008.