Graph-based Data Mining in Dynamic Networks: Empirical Comparison of Compression-based and Frequency-based Subgraph Mining

Chang Hun You, Lawrence B. Holder and Diane J. Cook
School of Electrical Engineering & Computer Science
Washington State University
Pullman, WA, USA
{changhun, holder, cook}@eecs.wsu.edu

Abstract

We propose a dynamic graph-based relational mining approach using graph-rewriting rules to learns patterns in networks that structurally change over time. A dynamic graph containing a sequence of graphs over time represents dynamic properties as well as structural properties of the network. Our approach discovers graph-rewriting rules, which describe the structural transformations between two sequential graphs over time, and also learns description rules that generalize over the discovered graph-rewriting rules. The discovered graph-rewriting rules show how networks change over time, and the description rules in the graph-rewriting rules show temporal patterns in the structural changes. We apply our approach to biological networks to understand how the biosystems change over time. Our compression-based discovery of the description rules is compared with the frequent subgraph mining approach using several evaluation metrics.

1. Introduction

Most of our world can be represented as networks including entities and relationships between the entities. For example, biological cells can be represented as biological networks, which include various molecules and relationships between molecules. The biological cells are also active systems, which promote reproduction and sustain its life. Active systems refer to dynamic properties of biological networks, which continuously change, while a cell performs various biological activities. Like the analysis of biological networks, analysis of dynamic networks is an emergent area of graph-based data mining. The dynamic graph contains a sequence of graphs representing a network changing over time. We propose a novel approach to analyze structural features along with temporal features in a dynamic graph.

For analysis of dynamic graphs, we first structurally represent how one graph is transformed into another. Several approaches [4] can measure the difference between two graphs. But our first goal is to describe how two graphs are different, not merely that they are different or by how much. We use graph-rewriting rules to show how two graphs are different. The second step is to discover temporal patterns in the discovered graph-rewriting rules. The temporal patterns can describe how and which graph-rewriting rules are applied in order to generate a sequence of changing graphs over time.

This paper first introduces several preceding approaches related to analysis of dynamic graphs. Then, we present our definition of graph-rewriting rules and more general description rules. We also introduce our two step algorithm to discover graph-rewriting rules in a dynamic graph, and description rules in the discovered graph-rewriting rules using a compression-based approach. In our experiments, we generate several dynamic graphs of the biological networks using the mathematical modeling data. Then, we apply our approach to the dynamic graphs. We compare our compression-based approach with the frequent subgraph mining approaches [17, 28] in learning the description rules from the graph rewriting rules. The description rules show which graph rewriting rules are repeated periodically and can help us predict the future changes of the dynamic graphs.

2. Related Work

Several methods have addressed data mining in a dynamic graph. Sun et al. [26] propose a technique to discover communities and detect changes in graphs changing over time using matrix and encoding schemes. Other work [3, 4] proposes several detection measures of abnormal changes in the sequence of graphs and graph distance measures be-

978-0-7695-3503-6/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDMW.2008.68

tween two graphs. Lahiri et. al. [18] introduce an approach to predict the future structure in a dynamic network using frequent subgraphs. Our approach uses the compression-based metric to discover temporal patterns in a dynamic graph.

The graph is an abstract data structure consisting of vertices and edges which are relationships between vertices. Graph-based data mining denotes a collection of algorithms for mining the relational aspects of data represented as a graph. Graph-based data mining has two major approaches: frequent subgraph mining and compression-based approach. Compression-based approach will be described in section 4. Frequent subgraph mining is focused on finding frequent subgraphs in a graph. There are two well-known approaches to bioinformatics domains. Frequent SubGraph discovery, FSG, finds all connected subgraphs that appear frequently in a set of graphs. FSG starts by finding all frequent single and double edge graphs. During each iteration FSG expands the size of frequent subgraphs by adding one edge to generate candidate subgraphs [17]. Graph-based Substructure Pattern Mining, gSpan, uses the depth-first search and lexicographic ordering [28]. Koyuturk et. al. [14] applied the frequent subgraph mining approach to metabolic pathways.

In this work, we mainly use the compression-based approach to discover graph rewriting rules and description rules. To evaluate our approach, we compare our methods with the frequent subgraph mining approach.

Temporal data mining attempts to learn temporal patterns in sequential data, which is ordered with respect to some index like time stamps, rather than static data [23]. Temporal data mining is focused on discovery of relational aspects in data such as discovery of temporal relations or cause-effect association. In other words, 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. For example, they can identify temporal patterns of appearance of genes such that a gene, YBR218C, appears before another gene, YGL062W, but cannot identify how these two genes interact with each other.

Biological networks have various molecules and relations between them including reactions and relations among genes and proteins. In addition to the structural aspect, we also consider the temporal aspect of biological networks, because the biosystems always change their properties and structures while interacting with other conditions. Two approaches have been developed for the analysis of biological networks. One approach is graph-based data mining [15, 29]. This approach represents biological networks as graphs, where vertices represent molecules and edges represent relations between molecules, and discovers frequent

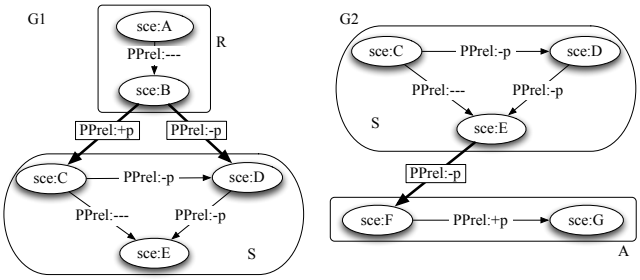

Figure 1. An instance of graph rewriting rules between graph G1 and G2 in a synthetic biological networks.

patterns in these graphs. Many approaches of graph-based data mining discover structural features of biological networks, but they overlook temporal properties. The other approach is mathematical modeling, which is an abstract model to describe a system using mathematical formulae [9, 13, 19, 20]. Most of these approaches, as a type of quantitative analysis, model the kinetics of pathways and analyzes the trends in the amounts of molecules and the flux of biochemical reactions. But most of them disregard relations among multiple molecules.

In our approach we combine dynamic graph analysis and temporal data mining to analyze the change in biological networks over time. The time-slice graphs of particular biological networks are obtained from the mathematical models of the networks.

3. Problem Definitions

3.1. Graph Rewriting Rules

This paper focuses on temporal and structural analysis of a dynamic graph. Our dynamic graph-based relational learning approach discovers graph rewriting rules in a series of graphs changing their structures over time. Each graph rewriting rule represents topological changes between two sequential graphs. Here, we define graph rewriting rules for our approach.

Graph rewriting is a method to represent topological changes of graphs using graph rewriting rules [8, 24]. Generally, graph rewriting rules identify subgraphs in a graph and modify them. Each graph rewriting rule defines a transformation between L and R, where L and R are subgraphs in two graphs G and H respectively, such that L is replaced by R, L is deleted, or R is created [21]. There are also several algorithms to discover the node or edge replacement graph grammar using the minimum description length principle [12, 16]. However, their scope is limited to static graphs.

Traditional approaches to the identification of graph rewriting rules determine which subgraphs will be replaced by other subgraphs. Our approach is focused on representing changing structures between two graphs rather than just what subgraphs change. First, we discover maximum common subgraphs between two sequential graphs G_1 and G_2. While this task is NP-Complete, we use a tractable heuristic approach described later. Then, we derive removal substructures from G_1 and addition substructures from G_2. Figure 1 shows an instance of this process. A maximum common subgraph (denoted by S) is discovered between two graphs, G_1 and G_2. Then the remaining structure in G_1 and G_2 becomes the removal (denoted by R) and addition (denoted by A) subgraph respectively.

Our graph-rewriting rules 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 1 represent the connection edges between G_1 (G_2) and removal rule R (addition subgraph A). The connection edges are important for two reasons. First, the connection edges show how the subgraphs are connected to the original graphs. Second, there can be more than one connection edge linking one subgraph to the original graph. Thus, the connection edges are significant features of graph-rewriting rules.

For this approach, we define several preliminary terms. A directed graph G 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, $L_v(V)$ is a function to assign labels to vertices and $L_e(E)$ is a function to assign labels to edges. An edge e ($\in E$) is directed from x to y as $e = (x, y)$, where $x, y \in V$.

Here, we define a dynamic graph DG as a sequence of n graphs as $DG = \{G_1, G_2, \cdots, G_n\}$, where each graph G_i is a graph at time i for $1 \leq i \leq n$. Then, we define a set of removal subgraphs RG and a set of addition subgraphs AG as follows.

$$RG_i = G_i / S_{i,i+1},$$
$$AG_{i+1} = G_{i+1} / S_{i,i+1}$$

RG_i denotes a set of removal subgraphs in a graph G_i, AG_{i+1} denotes a set of addition subgraphs in the next graph G_{i+1}, and $S_{i,i+1}$ is a maximum set of common subgraphs between two sequential graphs G_i and G_{i+1} in a dynamic graph DG.

A prior graph G_i is transformed to a posterior graph G_{i+1} by application of a set of graph rewriting rules $GR_{i,i+1}$ as denoted by

$$G_{i+1} = G_i \bigoplus GR_{i,i+1}$$

A set of graph rewriting rules $GR_{i,i+1}$ between two sequential graphs G_i and G_{i+1} is defined as follows.

$$\begin{aligned} GR_{i,i+1} &= \{(m, p, CE_m, CL_m), \cdots, \\ &\quad (n, q, CE_n, CL_n), , \cdots, \} \end{aligned}$$

m and n are indices of graph rewriting rules in a set $GR_{i,i+1}$. p and q are indices of a removal substructure in RG_i and an addition substructure in AG_{i+1} respectively. CE and CL are defined as a set of connection edges and a set of labels of the connection edges. Each element of RG and AG corresponds to a set of CE and CL, unless a removal (addition) substructure does not connect to the G_i (G_{i+1}). CE_k and CL_k represent connections between substructures and the original graphs ($k = m$ or n) as follows.

$$\begin{aligned} CE &= \{(d, X, Y), \cdots\} \\ CL &= \{label_{xy}, \cdots\} \end{aligned}$$

where d represents whether the edge is directed or undirected using d and u. X and Y denote the starting and ending vertices of the edge. Because the connection edge links the substructure to the original graph, one end of this edge is from the substructure and the other is from the original graph. The end vertex from the substructure starts with "s" followed by the index of the vertex, and the end vertex from the original graph starts with "g" followed by the index of the vertex. For example, $(d, g1, s3)$ represents the directed edge from a vertex 1 in the original graph to another vertex 3 in the substructure. $label_{xy}$ represents a label for the corresponding connection edge between two vertices X and Y. The number of elements of CE (CL as well) represents the number of connections between substructures and the original graph. If a substructure is not connected to the original graph, both sets of CE and CL are empty.

Using the definitions of graph rewriting rules, we describe more detail about the example in figure 1. As described previously, the graph rewriting rule, $GR_{1,2}$, includes one removal rule and one addition rule. The removal rule includes one substructure (denoted by R) and two connection edges (with boxed labels). The addition rule includes one substructure (denoted by A) and one connection edge (with a boxed label). G_1 is transformed to G_2 by application of $GR_{1,2}$ ($G_2 = G_1 \bigoplus GR_{1,2}$). $GR_{1,2}$ is described as follows,

$$\begin{aligned} GR_{1,2} &= \{(r_1, rSub_1, \{(d, s2, g3), (d, s2, g4)\}, \\ &\quad \{PPrel : +p, PPrel : -p\}), \\ &\quad (a_1, aSub_2, \{(d, g3, s1)\}, \{PPrel : -p\})\}, \end{aligned}$$

where r_1 represents an index into the set of removal rules and a_1 represents an index into the set of addition rules. A removal rule r_1 includes a removal substructure $rSub_1$ denoted by R in figure 1. $rSub_1$ was connected to the original graph G_1 by two edges $(d, s2, g3)$ and $(d, s2, g4)$, which are

labeled by $PPrel : +p$ and $PPrel : -p$. These connection edges are directed edges (indicated by 'd'). These two edges are connected from the substructure (denoted by 's') to the original graph (denoted by 'g'), where each number denotes a vertex number in the substructure or the original graph. For example, $(d, s2, g3)$ denotes a connection from a vertex number 2 in the substructure to a vertex number 3 in the original graph. In a similar way, an addition rule a_1 includes an addition substructure $aSub_1$ (denoted by A in figure 1), which is connected by one connection edge $(d, g3, s1)$ labeled by $PPrel : -p$.

The graph rewriting rules show how two sequential graphs are structurally different. After collecting all sets of graph rewriting rules in a dynamic graph, we also discover description rules in the graph rewriting rules, which can describe how the graphs change over time as well as what structures change.

3.2. Description Rules

As dynamic graph DG has n graphs, we have $n-1$ RGs and AGs after learning graph rewriting rules. Then, we learn *description rules* in the learned graph rewriting rules to describe the temporal patterns in dynamic graph. The *description rules* can be various forms based on data represented by dynamic graphs. Here, we propose one simple *description rule* DR, which represents repeated additions and removals (or vice versa), as follows.

$$DR = Sub(+t_a, -t_r)$$

Sub represents a subgraph, which adds to and 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 removals. For instance, $Sub\_a(+3, -2)$ denotes a repeated description rule with the addition after 3 time intervals from the last removal and the removal after 2 time intervals from the last addition. Naturally, there are other forms of *description rules* rather than the repeated add/remove rules. But this research focuses on the repeated rules as a starting point.

4. Approach

This section describes our approach to analyze dynamic graphs. We present a two step algorithm: Learning Graph Rewriting Rules and Learning Description Rules. The first algorithm learns graph rewriting rules in a dynamic graph to represent how two sequential graphs are different, in other words, how one graph is changed to the other graph. The second algorithm learns the repeated description rules in the learned graph rewriting rules to describe how the graphs in the dynamic graph change over time.

Our approach extends Cook and Holder's earlier work [5, 6], which is a graph-based relational learning approach to discover subgraphs. Their approach evaluates discovered subgraphs using the Minimum Description Length (MDL) principle to find the best subgraphs which minimize the description length of the input graph after being compressed by the subgraphs. 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 tries to minimize the $Compression$ of the graph as follows.

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

Their approach, which is called as $DiscoverSub()$ in our algorithms, tries to maximize the $Value$ of the subgraph, which is simply the inverse of the $Compression$. We present the $Value$ with our learned subgraphs in the results section.

In addition to the MDL based approach, $DiscoverSub()$ can use the size-based compression as follow,

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

where the size of a graph G is calculated as $size(G) = |V| + |E|$. Size is a less accurate measure of compression, as it does not account for the compression of vertex and edge label information like MDL. However, size is faster to compute, and results based on size are typically consistent with those of MDL.

In this research, we use both methods to discover subgraphs and compare our results with the frequent subgraph mining approach.

4.1. Learning Graph Rewriting Rules

This section describes our learning graph rewriting rules algorithm that discovers graph rewriting rules in a dynamic graph. The algorithm starts with a dynamic graph DG consisting of a sequence of n graphs as shown in algorithm 1. First, the algorithm creates a list of n virtual graphs, VGL, corresponding to n time series of graphs at line 1. Our approach uses a virtual graph to specify the application locations of graph rewriting rules. Because a graph may have multiple graph rewriting rules and several same-labeled vertices and edges, the exact locations of connection edges and rewriting rules are important to reduce the error. The next procedure is to create a two-graph set, $Graphs$, including two sequential graphs G_i and G_{i+1} (line 5) and to specify the $limit$ based on unique labeled vertices and

edges of G_i and G_{i+1} (line 6). UVL and UEL denote the number of unique vertex labels and edges in G_i and G_{i+1}. The $Limit$ specifies the number of substructures to consider when searching for a common substructure (line 6). The $Limit$ based on the number of labels in the input graph bounds the search for a common subgraph within polynomial time but ensure consideration of most of the possible subgraphs.

Algorithm 1: Learning Graph Rewriting Rules Algorithm

Input: $DG = \{G_1, G_2, \cdots, G_n\}$
Output: RRL

1 Create $VGL = \{VG_1, VG_2, \cdots, VG_n\}$
2 $RRL = \{\}$
3 **for** $i = 1$ *to* $n - 1$ **do**
4 $RemSubSet$ = $AddSubSet$ = $ComSubSet$ = $\{\}$
5 $Graphs = \{G_i, G_{i+1}\}$
6 $Limit = UVL + 4(UEL - 1)$
7 **while** *No more compression* **do**
8 BestSub = DiscoverSub(Limit, Graphs)
9 **if** $BestSub \in G_i$ & G_{i+1} **then**
10 Add $BestSub$ into $ComSet$
11 **end**
12 Compress $Graphs$ by $BestSub$
13 Mark $BestSub$ on VG_i and VG_{i+1}
14 **end**
15 Get $remSubs$ and CE from VG_i
16 Add $remSubs$ into $RemSubSet$ and CE into $RemCESet$
17 Get $addSubs$ and CE from VG_{i+1}
18 Add $addSubs$ into $AddSubSet$ and CE into $AddCESet$
19 Create RR from $RemSubSet$, $AddSubSet$, $RemCESet$, $AddCESet$
20 Add RR into RRL
21 **end**

The inner loop (lines 7 to 14) represents the procedure to discover common substructures between two sequential graphs. As described previously, $DiscoverSub()$ is used to find the maximum common subgraph. Even though to find the maximum common subgraph is NP-Complete, $DiscoverSub()$ can be used as a polynomial-time approximation to this problem using $Limit$ and $iteration$ as described later in section 4.3. After discovery of the best substructure, the algorithm checks whether the substructure is a subgraph of both graphs G_i and G_{i+1}. In the affirmative case, the best substructure is added into $ComSubSet$ and the two target graphs are compressed by replacing the substructure with a vertex. If the best substructure does not belong to one of the two graphs, the algorithm just compresses the graphs without adding any entry into $ComSubSet$. After compression, the algorithm discovers another substructure at the next iteration until there is no more compression.

Using the complete list of common substructures, $ComSubSet$, the algorithm acquires removal substructures, $remSubs$, and addition substructures, $addSubs$, (lines 15 and 17). First, the algorithm identifies vertices and edges not part of common substructures and finds each disconnected substructure in G_i and G_{i+1} using the modified Breadth First Search (mBFS), which adds each edge as well as each vertex into the queues as visited or to be visited. The marked substructures in G_i and G_{i+1} are removal and addition substructures respectively. While mBFS searches these removal and addition substructures, it also finds connection edges, CE, as described previously. These edges are added into $RemCESet$ and $AddCESet$, where removal and addition substructures are added into $RemSubSet$ and $AddSubSet$ respectively (in lines 16 and 18). Using these rewriting substructures and connection edges, rewriting rules (RR) are created and stored into RRL (in lines 19 and 20).

4.2. Learning Description Rules

After discovering graph rewriting rules in a dynamic graph, we try to discover repeated rewrites, which we call description rules, in the learned graph rewriting rules to better understand how graphs change over time. As shown in Algorithm 2, first, we add all sets (RG and AG) of the learned subgraphs as examples into one set $ExSet$, where every odd position is filled with RG and every even position is filled with AG (lines 1 to 3). The $ExSet$ contains $2(n-1)$ examples, because we discovered $n-1$ graph rewriting rules. Note that each example (each RG or AG) contains one or more graphs, which may not be connected to each other. We then use $DiscoverSubs$ again to find common subgraphs within $ExSet$. While a frequent subgraph miner could be used for this step, we prefer subgraphs that are both larger and frequent, i.e., compress well. After the discovery of the common subgraphs, $ExSet$ is compressed by this subgraph, and the discovery process is iterated until no more compression is achieved or we reached a user-defined limit $Iter$ on the number of iterations. In case that the best subgraph at the latter iteration includes the best subgraph at the former iteration, the results can show hierarchical clusters as a lattice. More detail on the hierarchical clustering approach is described in [11].

In our approach, we calculate the temporal distance between two consecutive instances of the best-compressing subgraphs to describe how long the removal (or addition) occurs after the previous addition (or removal). As described in section 3.2, $Sub\_1(+3, -2)$ and $Sub\_2(+5, -3)$ represent two description rules including two common sub-

Algorithm 2: Learning Description Rules Algorithm

Input: $RemSubSet$, $AddSubSet$, $Iter$, $Limit$
Output: $BestSubSets$, $ListOfDist$

```
Create ExSet = {}
Add RemSubSet as odd position into ExSet
Add AddSubSet as even position into ExSet
while No more compression and Iter > 0 do
    BestSub = DiscoverSub(ExSet)
    Add BestSub into BestSubSets
    Calculate distance between instances of BestSub
    and Add into ListOfDist
    Compress ExSet by BestSub
    Iter = Iter -1
end
```

graphs with temporal distances discovered at the first and second iterations respectively. Later, we compare our compression-based discovery of description rules with the frequent subgraph mining approach using several metrics.

4.3. Complexity Issue

The main computational challenge of our algorithm is to discover maximum common subgraphs between two sequential graphs, because this problem is known to be NP-hard [10]. To avoid this problem, first we use the $Limit$ to restrict the number of substructures to consider in each iteration. The $Limit$ is computed using the number of unique labels of vertices and edges in graphs or can be defined by the user. Second, our algorithm does not try to discover the all common substructures at once. In each step, the algorithm discovers one common, connected substructure and iterates the discovery process until discovering the whole maximum common subgraphs. In our target domain, graphs that represent biological networks usually contain unique vertex labels, because each vertex label denotes the name of the molecule. The maximum common subgraph problem in graphs with unique vertex labels is known to have quadratic complexity [7]. Therefore, discovery of the exact graph rewriting rules is still feasible. However, there will be a tradeoff between exactness and computation time when analyzing very large graphs.

5. Data Sets and Experiments

5.1. Mathematical Modeling

To evaluate our approach, we prepare eight dynamic networks representing biological networks that structurally change over time. The eight biological networks are generated from the mathematical modeling data, which are downloaded from [1], such as MAP kinase pathway [13], glycolysis pathway in yeast [9], respiration metabolic oscillations [27], oscillations in excitable cells cite [19], cell cycle by growth factor [22], mammalian circadian oscillator [25], circadian rhythms in fruit fly [20] and cell cycle signaling network [30]. We generate a static graph representing the biological networks from the KEGG PATHWAY data [2], where vertices represent compounds, genes, enzymes, relations and reactions, and edges represent relationships between vertices. Here, we assume only genes or chemical compounds change over time. Only when the amount of gene or chemical compound is larger than a user-defined threshold is the gene or chemical compound shown in the graph in the mathematical modeling data experiment. We also assume that each molecule can be shown at most once at each time slice.

The mathematical modeling approach explores only numerical values, such as the concentration of molecules and the flux of reactions. We propose to combine the result of mathematical modeling and graphs for structural and temporal analysis. The results of the mathematical modeling data show the trends of amounts of molecules (genes or compounds). We normalize these concentrations from 0 to 1, because we are focused on trends of the changes and the concentrations of different molecules vary significantly.

We generate the static graphs representing the biological networks from the mathematical model, and use a threshold th to activate compounds. At each time, a compound or gene, which has more than th amount, is shown in the graph. The reactions between the molecules are shown in the graph, only when all related molecules are activated. In other words, every related molecule should be activated to place a reaction in a graph. We chose from 0.3 to 0.4 as our thresholds to represent structural changes in the networks maximally. The simulation time of the mathematical modeling is varied from 140 seconds to 7,000 seconds. In our experiments, we generate 51 time series for training and the following 20 time series for testing. Thus, each dynamic graph has 51 graphs for training and 20 graphs for testing.

5.2. Experiments

As we discussed in the previous section, one goal of this research is to compare two graph-based data mining approaches in learning description rules. For this comparison, we introduce two metrics to measure the approaches.

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.

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

where the covered AGs and RGs are the addition and removal graphs in $ExSet$ that contain $BestSub$. These graphs are efficiently identified during the discovery of $BestSub$. $Coverage$ represents the portion of the learned subgraphs (the removal or addition subgraphs) described by description 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 RG_1 is 10, RG_2 is 12, AG_1 is 10, and AG_2 is 15. Also assume the $BestSub$ is found in RG_1 and AG_1 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 portion of) changes. Currently, $Coverage$ does not care about the size of connection edges ($|CE|$). Unless the size of 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 approaches as follows.

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

where P is the set of positions where we predict the $PredictedSub_i$ will show up, and $RealSub_i$ is the actual subgraph found at position i. $Prediction$ represents how much the predicted subgraph covers the discovered subgraphs in the testing experiments. For example, suppose we predict a subgraph s ($size(s) = 10$) will be shown 3 times in the testing data. Then, we discover the subgraph rs that is a subgraph of s ($size(rs) = 5$) at one time point, and the whole subgraph s at another time point. $Prediction$ is computed as $\frac{5/10+10/10+0/10}{3} = 0.5$. Currently, our $Prediction$ measure does not include the metric for the temporal prediction, i.e., when is the exact time the subgraph appears. We will discuss this issue in the next section.

We apply five different methods to compare two graph-based data mining approaches: compression-based and frequent subgraph mining approaches. We use MDL and size-based compression methods for the compression-based approach, and use three methods: size-based compression, most frequent, and largest patterns for the frequent subgraph mining. The size-based compression is computed as, $\max_i(size(Sub_i)num(Sub_i))$, where $num(Sub_i)$ denotes the number of instances of the subgraph Sub_i. The most frequent pattern denotes the largest subgraphs in the most frequent subgraphs, and the largest subgraphs denotes the largest subgraphs in all discovered subgraphs. The minimum support for the subgraphs is 4.0%, because we desire the subgraphs to appear more than two times in each removal and addition rule.

Our approach first learns graph rewriting rules (100 sets for each removal and addition) in the dynamic graph. Then, Algorithm 2 discovers the best subgraphs in the learned graph-rewriting rules as description rules using the above five methods. We will compare these approaches based on two metrics: $Coverage$ and $Prediction$. In addition to this quantitative assessment, we also visualize the description rules to better understand the dynamic graph.

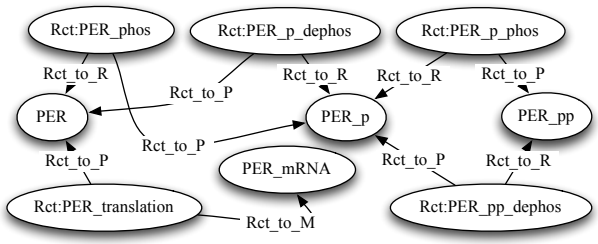

Figure 2. An instance of the best subgraph discovered in the experiment of the model 171.

6. Results and Discussion

6.1. Results

Table 1 shows the results of our experiments. Running time of these experiments is only a few seconds for each method because the graph size is small such as less than 100 at each time and 2,000 for each dynamic graph. The mathematical modeling data usually contains small numbers of molecules (less than 50). However, our approach can be applied to larger dynamic graphs that are generated based on microarray data, and the size of them are around 400 at each time (7,000 for dynamic graphs). In case that two or more methods have the same $Coverage$ and $Prediction$, they discover the same subgraph. As shown in Table 1, the compression-based approach is better or similar to the frequent subgraph mining approach in terms of $Coverage$ and $Prediction$ except two cases. In all cases, Prediction is greater than 70%. However, there are several challenges for the successful prediction. We will discuss these issues in the discussion section. In most cases, higher-valued subgraphs discovered using the compression-based approach can show higher $Coverage$ in the graph rewriting rules and description rules. Only two cases show the frequency-based approach outperforming the compression-based approach, because these two models show regularity in terms of only small subgraphs, not relatively larger ones, i.e., after one subgraph is added (or removed), portions of the subgraph are removed (or added) in other ways. We will discuss this issue as well in the discussion section.

Figure 2 shows an instance of the best subgraph discovered in the experiment of the model 171. The instances are discovered at time 11, 23, 35 and 47 as removals, and 6,18,

Table 1. Coverage of the best subgraphs in eight dynamic graphs, and prediction in the testing data. Name denotes the name of the mathematical model in [1] for each dynamic graph. Comp.-MDL and Comp.-Size denote the MDL and size-based compression methods. Freq.-Size, Freq.-MF and Freq.-Largest denote the size-based compression, the largest subgraph in the most frequent subgraphs, and the largest subgraph in the frequent subgraph mining approach. Cover. and Predict. denote coverage and prediction. The best coverage values are in bold.

Name	Comp.-MDL		Comp.-Size		Freq.-Size		Freq.-MF		Freq.-Largest	
	Cover.	Predict.	Cover.	Predict.	Cover.	Predict.	Cover.	Predict.	Cover.	Predict.
Model 10	0.025	1.0	0.050	1.0	0.050	1.0	**0.091**	1.0	0.050	1.0
Model 42	0.073	1.0	**0.099**	1.0	**0.099**	1.0	0.097	1.0	0.073	1.0
Model 90	**0.043**	0.900	0.042	1.0	**0.043**	0.900	0.031	1.0	**0.043**	1.0
Model 99	**0.127**	0.744	**0.127**	0.744	**0.127**	0.744	0.087	1.0	0.053	1.0
Model 110	**0.096**	1.0	**0.096**	1.0	**0.096**	1.0	0.077	1.0	0.090	1.0
Model 168	0.051	0.758	0.051	0.758	0.051	0.758	**0.062**	1.0	0.051	0.758
Model 170	0.040	1.0	**0.054**	0.778	0.040	1.0	**0.054**	0.778	0.040	1.0
Model 171	**0.073**	1.0	**0.073**	1.0	**0.073**	1.0	0.043	1.0	**0.073**	1.0

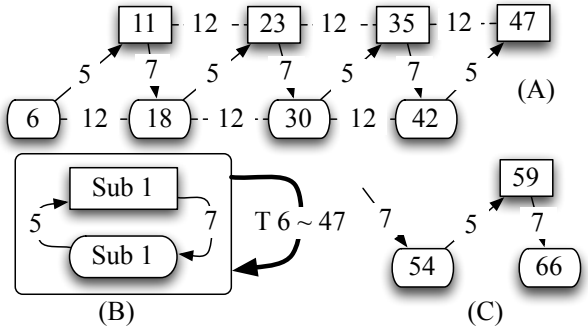

Figure 3. (A) Visualization of the graph rewriting rules including the subgraph in figure 2. (B) The description rule of the graph rewriting rules in (A). (C) Predicted graph rewriting rules from (B).

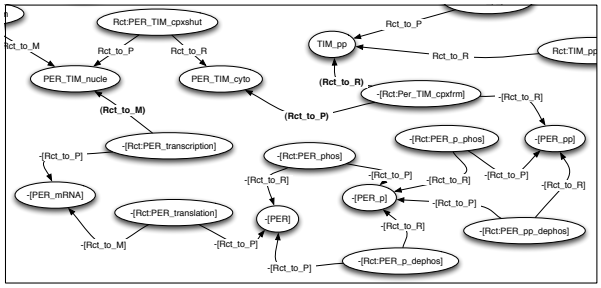

Figure 4. A visualization of discovered substructures of removal rule including the subgraph in figure 2. Labels marked by "-[]" represent the removal rules and labels marked by "()" represent the connection edges.

30 and 42 as additions as shown in figure 3 (A). The above rectangles in the figure represent the removals at the denoted time slice and the below eclipses represent the additions at the denoted time slice. The temporal distance from the removal to addition is 5, and from the addition to removal is 7 as shown in figure 3 (A). From the discovered subgraphs in figure 3 (B), we can learn the description rule $Sub_1(-5, +7)$ as shown in figure 3 (B). From the learned description rule, we can predict the future changes of the graph. As noticed, our research is focused on the prediction of the future structural changes, not the structure of the graph at the specific time. We predict three graph rewriting rules, i.e., three structural changes, at time 54, 59, and 66. As we predict, the instances of the subgraph in figure 2 are added at time 54 and 66, and removed at time 59.

As described previously, we use graph rewriting rules to represent the structural changes between two sequential graphs. Figure 4 shows a visualization of the graph rewriting rule including the best subgraph in figure 2. The removal subgraph including the best subgraph are shown with the labels marked by "-[]". In case of addition rules, the labels are marked by "+[]". The connection edges are represented with the labels marked by "()". Our graph-rewriting rules also show us how the subgraphs are related to the original graph using the connection edges as follows.

$$\begin{aligned} GR_{11,12} &= \{r_1, rsub_1, CE, CL\} \\ CE &= \{(d, s2, g10), (d, s11, g5), (d, s11, g9)\} \\ CL &= \{Rel\_to\_M, Rct\_to\_R, Rct\_to\_P\} \end{aligned}$$

These results are biologically significant, because they describe the repeated structural changes in the biological networks. The mathematical researches [9, 13, 19, 20, 22, 25, 27, 30] describe that there are periodic changes of molecules (i.e., concentrations) to perform the metabolism

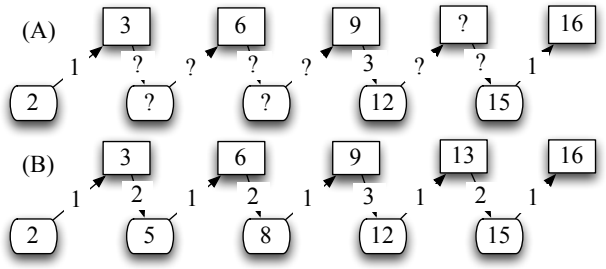

Figure 5. Visualization of the graph rewriting rules the best subgraph discovered in the model 99 using the MDL method (A) and the Freq.-MF method (B).

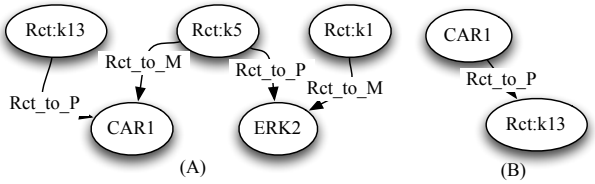

Figure 6. (A) An instance of the subgraph in figure 5 (A) discovered in the model 99 using the MDL method. (B) An instance of the subgraph in figure 5 (B) discovered in the model 99 using the Freq.-MF method.

or other processes in a cell. Our results show known patterns in the temporal changes. In addition, the results shows structural changes, but not merely changes of amount. The mathematical modeling [20] shows the trends of changing amounts of two proteins (PER and TIM) and related molecules, where they show periodic increase and decrease. Figure 3 (A) shows the graph rewriting rules including PER protein and related molecules that correspond to periodic changes in the amount of the molecule. In addition to the change of one element, our results 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 (figure 2) and how the subgraphs are linked to the original graphs (figure 4). Our results show patterns in the structural changes, not merely changes of amount.

6.2. Discussion

Even though our results can predict the future structural changes of dynamic networks and represent how the changes are related to other elements, there are still several challenges to overcome.

The first challenge is to synchronize the temporal patterns with the structural patterns. As we discussed in the experiment section, each molecule can appear only once at each time slice in our experiments. For this reason, a subgraph (or molecule) cannot be added (removed) successively, without any removal (addition). However, the current approach can discover the successive removal or addition as shown in figure 5. Figure 5 (A) shows several successive removals and additions. The question marks denote the time where the subgraph is not discovered, but should be discovered. The subgraph is not discovered because portions of the subgraph are removed (or added) in other ways, i.e., included into the other subgraphs.

Figure 6 shows an interpretation of this problem. Some elements of the subgraph in figure 6 (A) are included in the subgraph in figure 6 (B). The instances of the subgraph in figure 6 (B) are discovered in the normal way as shown in figure 5 (B). In other words, some instances of the subgraph in figure 6 (A) are removed and added separately (at different times). To overcome this problem, we need an approach to discover the subgraphs over the multiple time slices.

The next challenge is to evaluate the predicted time when the subgraph will appear. This issue could be addressed based on the domain. In the case of biological networks, the order of the structural changes is more important than the exact time when the changes occur. But there might be some domain in which we need to predict the exact time a structural change will occur (i.e., social network analysis to predict the emergence of a possible threat).

Dynamic network analysis is important and necessary not only for biological network analysis, but also for many other domains, such as social networks, web mining and so on. There are also many challenges to overcome more than we have addressed here. Dynamic graph-based relational learning using graph rewriting rules is focused on representing the changes between two graphs (i.e., two states) rather than the graph at the specific time. But analyzing the changes and predicting future changes of the structure of networks are good starting points for analysis of dynamic networks.

7. Conclusion

This research formalizes graph rewriting rules to describe structurally changing networks and describes an algorithm to discover graph rewriting rules in a dynamic graph. Our second step is to learn the general description rules in the discovered graph rewriting rules. If the data show the repeated removals and additions, our approach can discover the novel description rules describing how the graphs in a dynamic graph change over time. The algorithm is evaluated with the dynamic graphs representing biological networks in combination with the mathematical modeling data. We also compare our compression-based discovery of the description rules with the frequent subgraph mining approach. The results show that the frequency-based

approach, while typically faster, can miss higher-valued patterns discovered using the compression-based approach.

The graph-rewriting rules of dynamic networks can describe how the networks change over time. The graph-rewriting rules with the connection edges show how the learned subgraphs in the rewriting rules are related to the original graphs. The learned description rules in the graph rewriting rules can describe not only structural changes of networks but also temporal patterns in the series of the structural changes. Our approach can also help us to visualize the removal and addition of subgraphs at each time to show how the graphs structurally change. Our approach helps us to better explore how networks change over time and guides us to understand the structural behaviors of the dynamic network. In the biological network domain, the temporal patterns in structural changes of the network under specific conditions (e.g., infection) can provide essential information for drug discovery or disease treatments.

The future works follow several directions. First, we need a better approach to learn description rules that can cover graph rewriting rules that are divided over several consecutive time slices. 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 occur. Our evaluation will also include regenerating a dynamic graph using the discovered graph rewriting rules and comparing it to the original dynamic graph from real world data.

References

[1] Biomodels database at European Bioinformatics Institute. *http://www.ebi.ac.uk/biomodels*.

[2] Kegg. *http://www.genome.jp*.

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

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

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

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

[7] P. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl. On graphs with unique node labels. In *Proceedings of the IAPR Workshop on GBR*, pages 13–23, 2003.

[8] H. Dörr. *Efficient Graph Rewriting and Its Implementation*. Springer, 1995.

[9] K. N. et. al. Sustained oscillations in glycolysis: an experimental and theoretical study of chaotic and complex periodic behavior and of quenching of simple oscillations. *Biophysical Chemistry*, 7:49–62, 1998.

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

[11] I. Jonyer, D. Cook, and L. Holder. Discovery and evaluation of graph-based hierarchical conceptual clusters. *Journal of Machine Learning Research*, 2:19–43, 2001.

[12] I. Jonyer, L. Holder, and D. Cook. Mdl-based context-free graph grammar induction. In *Proceedings of FLAIRS*, 2003.

[13] B. Kholodenko. Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades. *Eur J Biochem*, 267(6):1583–8, Mar 2000.

[14] M. Koyuturk, A. Grama, and W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 20:200–207, 2004.

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

[16] J. P. Kukluk, L. B. Holder, and D. J. Cook. Inference of node replacement recursive graph grammars. In *Proceedings of the SIAM International Conference on Data Mining*, 2006.

[17] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of the ICDM*, pages 313–320, 2001.

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

[19] M. Laub and W. Loomis. A molecular network that produces spontaneous oscillations in excitable cells of dictyostelium. *Mol Biol Cell*, 9(12):3521–32, Dec 1998.

[20] J. Leloup and A. Goldbeter. A model for circadian rhythms in drosophila incorporating the formation of a complex between the per and tim proteins. *J Biol Rhythms*, 13(1):70–87, Feb 1998.

[21] K. Nupponen. The design and implementation of a graph rewrite engine for model transformations. Master's thesis, Helsinki Univ. of Tech., Dept. of Comp. Sci. & Eng., 2005.

[22] M. Obeyesekere, S. Zimmerman, E. Tecarro, and G. Auchmuty. A model of cell cycle behavior dominated by kinetics of a pathway stimulated by growth factors. *Bull Math Bio*, 61(5):917–34, Sep 1999.

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

[24] G. Rozenberg. *Handbook of Graph Grammars and Computing by Graph Transformation*. World Scientific, 1997.

[25] S. B.-W. S, J. Wolf, H. Herzel, and A. Kramer. Modeling feedback loops of the mammalian circadian oscillator. *Biophys J*, 87(5):3023–34, Nov 2004.

[26] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the ACM SIGKDD*, pages 687–696, 2007.

[27] J. Wolf, H. Sohn, R. Heinrich, and H. Kuriyama. Mathematical analysis of a mechanism for autonomous metabolic oscillations in continuous culture of saccharomyces cerevisiae. *FEBS Lett*, 499(3):230–4, Jun 2001.

[28] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the ICDM*, pages 721–724, 2002.

[29] C. You, L. Holder, and D. Cook. Application of graph-based data mining to metabolic pathways. In *Proceedings of IEEE ICDM Workshop on Data Mining in Bioinformatics*, 2006.

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