Algorithms for Mining the Coevolving Relational Motifs in Dynamic Networks

REZWAN AHMED, University of Minnesota
GEORGE KARYPIS, University of Minnesota

Computational methods and tools that can efficiently and effectively analyze the temporal changes in dynamic complex relational networks enable us to gain significant insights regarding the entity relations and their evolution. This paper introduces a new class of dynamic graph patterns, referred to as coevolving relational motifs (CRMs), which are designed to identify recurring sets of entities whose relations change in a consistent way over time. Coevolving relational motifs can provide evidence to the existence of, possibly unknown, coordination mechanisms by identifying the relational motifs that evolve in a similar and highly conserved fashion. We developed an algorithm to efficiently analyze the frequent relational changes between the entities of the dynamic networks and capture all frequent coevolutions as CRMs. Our algorithm follows a depth-first exploration of the frequent CRM lattice and incorporates canonical labeling for redundancy elimination. Experimental results based on multiple real world dynamic networks show that the method is able to efficiently identify CRMs. In addition, a qualitative analysis of the results shows that the discovered patterns can be used as features to characterize the dynamic network.

Categories and Subject Descriptors: G.2.2 [**Graph Theory**]: Graph mining; I.2.8 [**Problem Solving, Control Methods, and Search**]: Graph mining

General Terms: Algorithms

Additional Key Words and Phrases: Dynamic networks, Evolving pattern mining, Network evolution

ACM Reference Format:
Rezwan Ahmed and George Karypis, 2013. Algorithms for Mining the Coevolving Relational Motifs in Dynamic Networks. *ACM Trans. Knowl. Discov. Data.* V, N, Article A (YYYY), 28 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Networks are generic models used in various applications from different domains to model the relations between various entities. Examples of some widely studied networks include the friends-networks of popular social networking sites like Facebook and Myspace [Boyd and Ellison 2007], the Enron email network [Borgwardt et al. 2006; Cohen 2005], co-authorship and citation networks [Liu et al. 2005; Lawrence et al. 1999; Chen and Redner 2010; Perc 2010], web-page linking networks [Brin and Page 1998; Liu 2007], protein-protein interaction networks [Hu 2005; Schwikowski et al. 2000], and networks derived from the IMDB movie database[Koren et al. 2007]. Many of these networks are dynamic in nature as the entities and relations that need to be modeled change over time. With the emergence of new application areas, there has been a great need to analyze the temporal changes underlying many of these systems, and to develop tools capable of capturing the dynamic aspects of the data, as well

Author's addresses: R. Ahmed and G. Karypis, Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455. Email: {ahmed,karypis}@cs.umn.edu.

DOI: http://dx.doi.org/10.1145/0000000.0000000

as analyze and process this data. Such analysis can identify relational patterns that provides strong observational evidence of the existence of mechanisms that control, coordinate, and trigger the evolutionary changes, which can be used to focus further studies and analysis. A recent review [Holme and Saramäki 2012] analyzed various dynamic networks and stated the need for tools to efficiently analyze and mine such networks. Additionally, a related body of research [Perc and Szolnoki 2010] presented the need for analyzing co-evolution in such networks while investigating the evolution of cooperation among humans in social dynamics.

Dynamic networks have recently emerged as an important research area. The focus of this growing research has been mainly defining important recurrent structural patterns and developing algorithms for their identification. In recent years, studies have been done for finding frequent patterns [Borgwardt et al. 2006], clustering [Chakrabarti et al. 2006], characterizing network evolution rules [Berlingerio et al. 2009], detecting related cliques [Cerf et al. 2009; Robardet 2009], identifying conserved relational states [Ahmed and Karypis 2012], finding subgraph subsequences [Inokuchi and Washio 2010], and identifying co-evolution patterns capturing attribute trends [Jin et al. 2007; Desmier et al. 2012] in dynamic networks. Although the existing techniques can detect various patterns in a dynamic network, they have not focused on identifying frequent conserved changes of the relational patterns over time. Understanding the evolution of relational patterns that occur frequently can provide evidence of, possibly unknown, coordination mechanisms and to the existence of external factors that are responsible for changing the stable relational patterns in the dynamic network.

Our contribution in this paper is two fold: Firstly, we introduce a new class of patterns, referred to as coevolving relational motifs (CRM), designed to capture patterns that change in a consistent way in a dynamic networks. CRMs identify consistent patterns of relational motif evolution that can provide valuable insights on the processes of the underlying networks. Secondly, we present an algorithm, referred to as CRMminer, to efficiently mine a subclass of these patterns by identifying all frequent coevolving relational motifs such that the motifs that make up the CRM share at least one relation (i.e., an edge) that changes over time. Our algorithm follows a depth-first exploration of the frequent CRM lattice and incorporates canonical labeling for redundancy elimination. We impose both frequency based and node overlap based constraints for pruning the search space to increase efficiency and an approximate pruning to reduce the complexity of our algorithm.

We provide a comprehensive evaluation of the performance of CRMminer and the usefulness of the discovered patterns. We experimentally evaluate the performance and scalability of CRMminer on a large co-authorship network, on a cell culture bioprocess network (multivariate time series data), and on a a market sales network (multivariate time series data) by varying different input parameters. Furthermore, our experiments show that the approximate version of CRMminer is able to identify the majority of the CRMs while reducing the amount of time that is required. In addition, we investigate some discovered coevolving relational motifs from all three datasets and provide a qualitative analysis of the information captured in them. We show that some of the discovered CRMs capture relational changes between the nodes that are thematically different (i.e., edge labels transition between two clusters of topics that have very low similarity). In addition, we investigate the extent to which these discovered CRMs can lead to high quality features to build a predictive model. Our results, in the context of a bioprocess dataset, show that the discovered CRMs are present mostly as part of the high yield production runs.

The rest of the paper is organized as follows. Section 2 reviews some graph-related definitions and introduces the notation that is used in the paper. Section 3 captures

a brief survey of the related research in this area. Section 4 introduces the concept of coevolving relational motifs and Section 5 describes in detail the algorithm to detect such patterns. Section 6 describes the datasets and metrics used in the experimental evaluation of our algorithm. Section 7 presents a detailed analysis of the experimental results and Section 8 provides some concluding remarks and future directions.

2. DEFINITIONS AND NOTATION

A relational network is represented via labeled graphs. A *labeled graph* $G = (V, E, L[V], L[E])$ is composed of a set of nodes V modeling the entities of the network, a set of edges E modeling the relations between these entities, a set of node labels $L[V]$ modeling the type of the entities ($|V| = |L[V]|$), and a set of edge labels $L[E]$ modeling the type of the relations ($|E| = |L[E]|$). The labels assigned to the vertices (edges) are typically not unique and multiple vertices (edges) can have the same label. A *subgraph* $G' = (V', E', L[V'], L[E'])$ of G is a graph such that $V' \subseteq V$, $E' \subseteq E \cap (V' \times V')$. An *induced subgraph* $G'' = (V'', E'', L[V''], L[E''])$ of G is a graph such that $V'' \subseteq V$, $E'' \subseteq E$ and $\forall (u, v) \in E$ such that $v \in V''$ and $u \in V''$, $(u, v) \in E''$.

Given a connected graph $G = (V, E)$ and a depth-first search (DFS) traversal of G, its *depth-first search tree* T is the tree formed by the forward edges of G. All nodes of G are encoded with subscripts to order them according to their discovery time. Given a DFS tree T of graph G containing n nodes, the root node is labeled as (v_0) and the last discovered node is labeled as (v_{n-1}). The *rightmost path* of the DFS tree T is the path from vertex v_0 to vertex v_{n-1}.

A *dynamic network* $\mathcal{N} = \langle G_1, G_2, \ldots, G_T \rangle$ is modeled as a finite sequence of graphs, where each $G_t = (V, E_t, L_t[V], L_t[E_t])$ is a labeled graph describing the state of the system at a discrete time interval t. The term *snapshot* will be used to refer to each of the graphs in the sequence. Snapshots are assumed to contain the same set of nodes, which will also be referred to as the nodes of $\mathcal{N}$, denoted by $V_{\mathcal{N}}$, but potentially different sets of edges and node/edge labels. When nodes appear or disappear over time, the set of nodes of each snapshot is the union of all the nodes over all snapshots. Also, the nodes across the different snapshots are numbered consistently, so that the ith node of G_k ($1 \leq k \leq T$) will always correspond to the same ith node of $\mathcal{N}$.

We define the *span sequence* of an edge as the sequence of maximal-length time intervals in which an edge is present in a consistent state. An edge (u, v) is in a *consistent state* over a maximal time interval $s{:}e$ if it is present in all snapshots $G_s, \ldots, G_e$ with the same label l and it is different in both G_{s-1} and G_{e+1} (assuming $s > 1$ and $e < T$). The span sequence of an edge will be described by a sequence of vertex labels, edge labels and time intervals of the form $\langle (l_{u_1}, l_{e_1}, l_{v_1}, s_1{:}e_1), \ldots, (l_{u_n}, l_{e_n}, l_{v_n}, s_n{:}e_n) \rangle$, where $l_{u_i}, l_{v_i} \in L[V], l_{e_i} \in L[E]$, $s_i \leq e_i$, and $e_i \leq s_{i+1}$.

An *injection* is defined as a function $f : A \rightarrow B$ such that $\forall a, b \in A$, if $f(a) = f(b)$, then $a = b$. A function $f : A \rightarrow B$ is a bijective function or *bijection*, iff $\forall b \in B$, there is a unique $a \in A$ such that $f(a) = b$. A function composition $g \circ f$ implies that for function $f : X \rightarrow Y$ and $g : Y \rightarrow Z$, then *composite* function $g \circ f : X \rightarrow Z$.

A *relational motif* is a subgraph that occurs frequently in a single snapshot or a collection of snapshots. In Figure 1, the three-vertex subgraph consisting of the shaded nodes that are connected via the labeled edges corresponds to a relational motif that occurs a total of four times (twice in G_1 and once in each of G_2 and G_3). The set of nodes that support the multiple occurrences of a relational motif do not need to be the same. In order to determine if a snapshot supports a relational motif (and how many times), we need to perform subgraph isomorphism operations (i.e., identify the embeddings of the relational motif's graph pattern). We will use M to denote a relational motif and the underlying subgraphs will be denoted by the tuple $(N, A, L[N], L[A])$, where N is

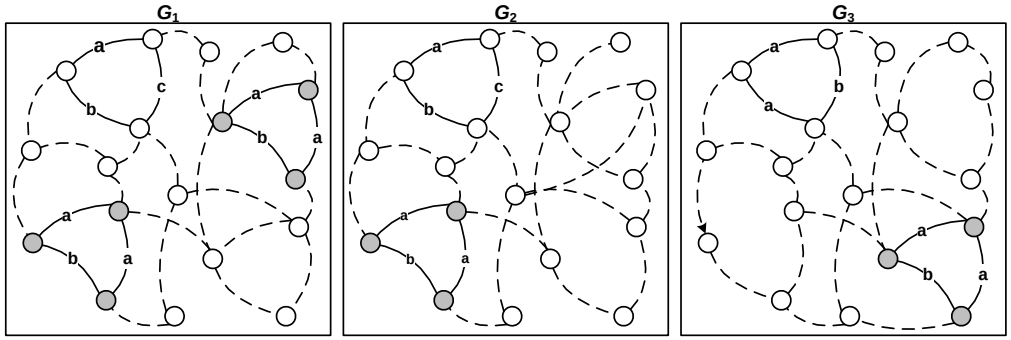


Fig. 1. Examples of relational motifs. The shaded nodes that are connected via the labeled edges corresponds to a relational motif.

the set of nodes, A is the set of edges (arcs), $L[N]$ is the set of node labels, and $L[A]$ is the set of edge labels.

3. RELATED WORK

Unlike the problem of finding patterns in static networks for which there exist a considerable body of research [Kramer et al. 2001; Pei et al. 2001b; Zaki 2002; Asai et al. 2002; Inokuchi et al. 2000; Huan et al. 2003; Kuramochi and Karypis 2004; Hu et al. 2005; Pei et al. 2005; Yan et al. 2005], the problem of mining dynamic networks is comparatively less studied.

Borgwardt et. al. [Borgwardt et al. 2006] introduced the notion of the *dynamic subgraph*, which extends the traditional notion of the subgraph to include the sequence of subgraphs that exist in a consecutive sequence of snapshots and developed algorithms to identify the set of dynamic subgraphs that occur frequently in a dynamic network. Jin et. al. [Jin et al. 2007] introduced the notion of the *trend motif*, which is a connected subgraph with nodes containing time series and each node's time series exhibits a consistent trend over a time interval. Their work focused only on developing algorithms for finding frequently occurring trend motifs that show either an increasing or a decreasing trend. With a similar objective, Desmier et. al. [Desmier et al. 2012] defined the problem of mining cohesive co-evolving patterns which capture the local co-evolution of similar vertices at several timestamps based on their attributes trends. Lahiri et. al. [Lahiri and Berger-Wolf 2008] proposed a mining problem of finding periodic or near periodic subgraphs in dynamic networks.

Berlingerio et. al. [Berlingerio et al. 2009] also provided an algorithm to detect frequent subgraphs in time-evolving graphs for deriving graph-evolution rules that satisfy a minimum confidence constraint. Robardet [Robardet 2009] represented the frequent patterns of a graph as *pseudo-cliques* and proposed an algorithm that first mines each graph snapshot of a dynamic graph for local patterns and then combines these with patterns from previous snapshot based on some constraints to form evolving patterns. Inokuchi et. al. [Inokuchi and Washio 2008; 2010] solved a similar problem of finding frequent induced subgraph subsequences from graph sequence data and capturing the changes of a subgraph over the subsequence. Ahmed et. al. [Ahmed and Karypis 2012] introduced a new class of patterns that captures the time-persistent relations among the nodes, called *relational states* and presented an algorithm to identify all maximal non-redundant evolution paths of the stable relational states of a dynamic network.

A related body of research has investigated the task of identifying and tracking patterns in biological networks [You et al. 2009; Koyutürk et al. 2006; Wackersreuther

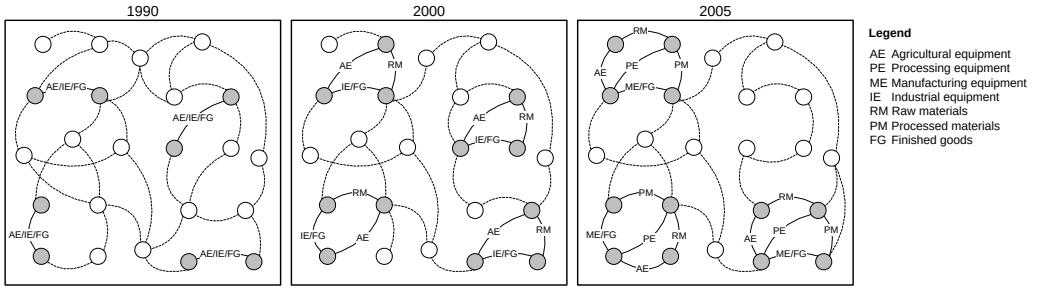


Fig. 2. An example of a coevolving relational motif in the context of a hypothetical country-to-country trading network where labels represent the commodities been traded.

et al. 2010], evolving communities in social networks [Berger-Wolf and Saia 2006; Duan et al. 2009], and evolutionary clustering [Chakrabarti et al. 2006; Tang et al. 2008]. Even though these methods provide valuable insights on the evolution of dynamic networks, the nature of co-evolving relational patterns focused in this paper is different from the various types of evolution addressed in other papers.

4. COEVOLVING RELATIONAL MOTIFS

Coevolving relational motifs are designed to identify the relational patterns that change in a consistent way over time. An example of this type of conservation is illustrated in Figure 2, in the context of a hypothetical country-to-country trading network where labels represent the commodities been traded. The network for 1990 shows a simple relational motif (M_1) between pairs of nodes that occurs four times (shaded nodes and solid labeled edges). This relational motif has evolved in the network for 2000 in such a way so that in all four cases, a new motif (M_2) that includes an additional node has emerged. Finally, in the network for 2005 we see that three out of these four occurrences have evolved to a new motif (M_3) that now involves four nodes. This example shows that the initial relational motif among the four sets of nodes has changed in a fairly consistent fashion over time (i.e., it *coevolved*) and such a sequence of motifs $M_1 \rightsquigarrow M_2 \rightsquigarrow M_3$ represents an instance of a CRM.

CRMs identify consistent patterns of relational motif evolution that can provide valuable insights on the processes of the underlying networks. For example, the CRM of Figure 2 captures the well-known phenomenon of production specialization due to economic globalization, in which the production of goods have been broken down into different components performed by different countries [Friedman 2005]. Similarly, CRMs in health-care networks can capture how the set of medical specialties required to treat certain medical conditions have changed over the years, in communication networks CRMs can capture the evolution of themes being discussed among groups of individuals as their lifestyles change, whereas CRMs in corporate email networks can capture how the discussion related to certain topics moves through the companies' hierarchies.

The formal definition of a CRM that is used in this paper is as follows:

Definition 4.1 *A CRM of length m is a tuple $\{N, \langle M_1, \ldots, M_m \rangle\}$, where N is a set of vertices and each $M_j = (N_j, A_j)$ is a relational motif defined over a subset of the vertices of N that satisfies the following constraints:*

i) it occurs at least ϕ times,
ii) each occurrence uses a non-identical set of nodes,
iii) $M_j \neq M_{j+1}$, and
iv) $|N_j| \geq \beta |N|$ where $0 < \beta \leq 1$.

A relational motif M_j is *defined* over a subset of vertices N if there is a injection ξ_j from N_j to N. An m-length CRM *occurs* in a dynamic network whose node set is V if there is a sequence of m snapshots $\langle G_{i_1}, G_{i_2}, \ldots, G_{i_m} \rangle$ and a subset of vertices B of V (i.e., $B \subseteq V$) such that:

i) there is a bijection ξ from N to B
ii) the injection $\xi \circ \xi_j$ is an embedding of M_j in G_{i_j}
iii) there is no embedding of M_j via the injection $\xi \circ \xi_j$ in $G_{i_{j+1}}$ or no embedding of M_{j+1} via the injection $\xi \circ \xi_{j+1}$ in G_{i_j}.

For example, the number of occurrences of the CRM in Figure 2 is 3. Note that the third condition in the above definition is designed to ensure that for each pair of successive motifs at least one of them is not supported by the snapshot-nodes pair that supported the other motif. This is done to ensure that there is a relational change between the nodes associated with those embeddings in each others snapshot.

The purpose of the parameters ϕ and β in Definition 4.1 are as follows: The parameter ϕ is used to eliminate sequences of evolving motifs that are not frequent enough to indicate the existence of an underlying process driving these changes. The parameter β is used to control the degree of change between the sets of nodes involved in each motif of a CRM and enforces a minimum node overlap among all motifs of CRM. Finally, the third constraint of Definition 4.1 limits the discovered CRMs to only an evolving sequence of motifs and not a sequence that remains the same. Note that the frequent dynamic subgraphs introduced by Borgwardt et. al. [Borgwardt et al. 2006] (Section 3) correspond to CRMs in which the snapshots supporting each set of nodes are restricted to be consecutive and $\beta = 1$.

In this paper we focus on developing an efficient algorithm to mine a subclass of the CRMs, such that in addition to the conditions mentioned in Definition 4.1, the motifs that make up the CRM, also share at least one edge that itself is a CRM. Formally, we focus on identifying the CRMs $c = \{N_c, \langle M_1, \ldots, M_m \rangle\}$ that contain at least a pair of vertices $\{u, v\} \in N_C$ such that each induced subgraph $M_i^{'}$ of M_i on $\{u, v\}$ is connected and $x = \{\{u, v\}, \langle M_1^{'}, \ldots, M_m^{'} \rangle\}$ is a CRM. A CRM like x that contains only one edge and two vertices will be called an *anchor*. We focus our CRM enumeration problem around the anchors, since they ensure that the CRM's motifs contain at least a pair of nodes in common irrespective of the specified overlap constraint that evolves in a conserved way. It also characterizes how the network around these core set of entities coevolved with them. We will refer to the class of CRMs that contain an anchor as *anchored CRMs*. For the rest of the discussion, any references to a CRM will assume it is an anchored CRM.

Given the above definition, the work in this paper is designed to develop efficient algorithm for solving the following problem:

Problem 1 *Given a dynamic network $\mathcal{N}$ containing T snapshots, a user defined minimum support ϕ ($1 \leq \phi$), a minimum number of edges $k_{\min}$ per CRM, and a minimum number of motifs $m_{\min}$ per CRM, find all CRMs such that the motifs that make up the CRM, also share an anchor CRM.*

A CRM that meets the requirements specified in Problem 1 is referred as a *frequent* CRM and it is *valid* if it also satisfies the minimum node overlap constraint (Definition 4.1(iv)).

5. FINDING COEVOLVING RELATIONAL MOTIFS

A consequent of the way anchored CRMs are defined is that the number of motifs that they contain is exactly the same as the number of motifs that exist in the anchor(s) that they contain. As a result, the CRMs can be identified by starting from all the

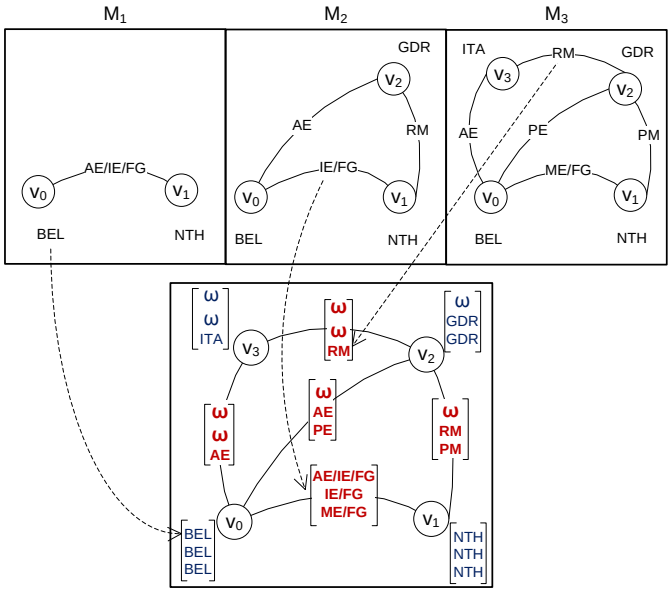


Fig. 3. A CRM Representation. The CRM c consists of 3 motifs $\langle M_1, M_2, M_3\rangle$ and represents relations among vertices $N = \{v_0, v_1, v_2, v_3\}$ using 5 edges. G_c (bottom graph) shows the CRM representation capturing vertex and edge label vectors.

available anchors and try to grow them by repeatedly adding edges as long as the newly derived CRMs satisfy the constraints and have exactly the same number of motifs as the anchor. Since a CRM can contain more than one anchor, this approach may identify redundant CRMs by generating the same CRM from multiple anchors. Therefore, the challenge is to design a strategy that is complete and non-redundant. To achieve this, we develop an approach that generates each CRM from a unique anchor. Given a CRM, the anchor from which it will be generated is referred to as its *seed anchor*.

The algorithm that we developed, named CRMminer, for finding all non-redundant valid CRMs (Problem 1) initially identifies all frequent anchors and then performs a depth-first exploration of each anchor pattern space along with a canonical labeling that is derived by extending the ideas of the minimum DFS code [Yan and Han 2002] to the case of CRMs for redundancy elimination. We impose frequency-based constraints by stopping any further exploration of a CRM when the pattern does not occur at least ϕ times in the dynamic network $\mathcal{N}$.

5.1. CRM Representation

A CRM $c = \{N, \langle M_1, \ldots, M_m\rangle\}$ is represented as a graph $G_c = (N, E_c)$, such that an edge $(u, v) \in E_c$ is a 5-item tuple $(u, v, \boldsymbol{l_u}, \boldsymbol{l_{u,v}}, \boldsymbol{l_v})$, where $u, v \in N$, the vectors $\boldsymbol{l_u}$ and $\boldsymbol{l_v}$ contain the vertex labels and $\boldsymbol{l_{u,v}}$ contains the edge labels of all motifs. If the CRM consists of m motifs, then $\boldsymbol{l_u} = \langle l_{u_1}, \ldots, l_{u_m}\rangle$, $\boldsymbol{l_v} = \langle l_{v_1}, \ldots, l_{v_m}\rangle$ and $\boldsymbol{l_{u,v}} = \langle l_{u_1,v_1}, \ldots, l_{u_m,v_m}\rangle$. The kth entry in each vector $\boldsymbol{l_u}$, $\boldsymbol{l_{u,v}}$, and $\boldsymbol{l_v}$ records the connectivity information among the vertices of the kth motif (M_k). If an edge (u, v) is part of motif M_k, then the kth entry of $\boldsymbol{l_u}$, $\boldsymbol{l_v}$, and $\boldsymbol{l_{u,v}}$ are set to the labels of u and v vertices, and the label of the (u, v) edge respectively. If both vertices u and v are part of motif M_k, but the (u, v) edge is not, or at least one of the vertices u or v is not part of the M_k motif (i.e., no (u, v) edge is possible), then ω is inserted at the kth entry of the $\boldsymbol{l_{u,v}}$ to capture the

disconnected state. Similarly, if u or v does not have any incident edges in the M_k motif (i.e., the vertices are not present in that motif), then ω is added as the vertex label at the kth entry of $\boldsymbol{l_u}$ or $\boldsymbol{l_v}$. Note that the value of ω is lexicographically greater than the maximum edge and vertex label.

This representation is illustrated in Figure 3. The CRM consists of 3 motifs $\langle M_1, M_2, M_3\rangle$ and represents relations among vertices N=$\{v_0, v_1, v_2, v_3\}$ using 5 edges. The edge between (v_0, v_1) exists in all 3 motifs capturing changes in relation as the edge label changes from $AE/IE/FG \rightsquigarrow IE/FG \rightsquigarrow ME/FG$. It is represented as $\boldsymbol{l_{v_0}}$=$\langle BEL, BEL, BEL\rangle$, $\boldsymbol{l_{v_1}}$=$\langle NTH, NTH, NTH\rangle$, and $\boldsymbol{l_{v_0,v_1}}$=$\langle AE/IE/FG, IE/FG, ME/FG\rangle$. The next edge between (v_1, v_2) appears in 2 motifs and the label vectors are represented as $\boldsymbol{l_{v_1}}$=$\langle NTH, NTH, NTH\rangle$, $\boldsymbol{l_{v_2}}$=$\langle \omega, GDR, GDR\rangle$, and $\boldsymbol{l_{v_0,v_1}}$=$\langle \omega, RM, PM\rangle$. Following similar process, we can represent edges (v_2, v_0), (v_2, v_3), and (v_3, v_0).

5.2. Mining Anchors

The search for CRMs is initiated by locating the frequent anchors that satisfy the CRM definition and the restrictions defined in Problem 1. This is done as following: Given a dynamic network $\mathcal{N}$, we sort all the vertices and edges by their label frequency and remove all infrequent vertices and edges. Remaining vertices and edges are relabeled in decreasing frequency. We determine the span sequences of each edge and list every edge's span sequence if that sequence contains at least a span with an edge label that is different from the rest of the spans. At this point, we use the sequential pattern mining technique prefixSpan [Pei et al. 2001a] to determine all frequent span sequences. Since the frequent sequences can be partial sequences of the original input span sequences, it is not guaranteed that they all contain consecutive spans with different labels. Thus, the frequent sequences that contain different consecutive spans in terms of label are considered as the anchors. The number of spans in a frequent span sequence corresponds to the total number of motifs in the anchor.

5.3. CRM Enumeration

Given an anchor c, we generate the set of desired CRMs by growing the size of the current CRM one edge at a time following a depth-first approach. To ensure that each CRM is generated only once in the depth-first exploration, we use an approach similar to the gSpan algorithm [Yan and Han 2002], which we have extended for the problem of CRM mining.

gSpan explores the frequent pattern lattice in a depth-first fashion. The pattern lattice is represented as a hierarchical search space where each node corresponds to a connected frequent pattern, the highest node being an empty pattern (i.e., a single vertex), the next level nodes represent 1-edge patterns, and so on. The nth level nodes, which represent n-edge patterns, contain one more edge than the corresponding $(n-1)$ level nodes. To ensure that each frequent pattern in this lattice is visited exactly once, gSpan's exploration amounts to visiting the nodes of the lattice (i.e., frequent patterns) by traversing a set of edges that form a spanning tree of the lattice. This spanning tree is defined by assigning to each node of the lattice a canonical label, called the minimum DFS code. A DFS code of a graph, is a unique label that is formed based on the sequence of edges added to that node during a depth-first exploration. The minimum DFS code, is the DFS code that is lexicographically the smallest. Given this canonical labeling, the set of lattice edges that are used to form the spanning tree correspond to the edges between two successive nodes of the lattice (parent and child) such that the minimum DFS code of the child can be obtained by simply appending the extra edge to the minimum DFS code of the parent. For example, given a DFS code $\alpha = \langle a_0, a_1, \cdots, a_m\rangle$, a valid child DFS code is $\gamma = \langle a_0, a_1, \ldots, a_m, b\rangle$, where b

is the new edge. This spanning tree guarantees that each node has a unique parent and all nodes are connected [Yan and Han 2002]. To efficiently grow a node, gSpan generates the child nodes by only adding those edges that originate from the vertices on the rightmost path of the DFS-tree representation of the parent node. It then checks whether the resulting DFS code of the child node corresponds to the minimum DFS code. Construction of the child nodes generated by adding other edges (i.e., not from the rightmost path) are skipped, since such child nodes will never contain a DFS code that corresponds to the minimum DFS code.

In order to apply the ideas introduced by gSpan to the problem of efficiently mining CRMs, we need to develop approaches for (i) representing the DFS code a CRM, (ii) ordering the DFS codes of a CRM using the DFS lexicographic ordering, (iii) representing the minimum DFS code of a CRM to use as the canonical label, and (iv) extending a DFS code of a CRM by adding an edge. Once properly defined, the correctness and completeness of frequent CRM enumeration follows directly from the corresponding proofs of gSpan.

5.3.1. DFS code of a CRM. In order to derive a DFS code of a CRM, we need to develop a way of ordering the edges. Given a CRM c, represented as a graph $G_c = (N, E_c)$, we perform a depth-first search in G_c to build a DFS tree T_c. The vertices (N) are assigned subscripts from 0 to $n-1$ for $|N| = n$ according to their discovery time. The edges (E_c) are grouped into two sets: the forward edge set $E_{T_c,fw} = \{(v_i, v_j) \in E_c |\ i < j\}$ and the backward edge set $E_{T_c,bw} = \{(v_i, v_j) \in E_c |\ i > j\}$. Let us denote a partial order on $E_{T_c,fw}$ as $\prec_{T_c,fw}$, a partial order on $E_{T_c,bw}$ as $\prec_{T_c,bw}$, and a partial order on E_c as $\prec_{T_c,bw+fw}$. Given two edges $e_1 = (v_{i_1}, v_{j_1})$ and $e_2 = (v_{i_2}, v_{j_2})$, the partial order relations are defined as:

a) $\forall e_1, e_2 \in E_{T_c,fw}$, if $j_1 < j_2$, then $e_1 \prec_{T_c,fw} e_2$.
b) $\forall e_1, e_2 \in E_{T_c,bw}$, if $i_1 < i_2$ or ($i_1 = i_2$ and $j_1 < j_2$), then $e_1 \prec_{T_c,bw} e_2$.
c) $\forall e_1 \in E_{T_c,bw}$ and $\forall e_2 \in E_{T_c,fw}$, if $i_1 < j_2$, then $e_1 \prec_{T_c,bw+fw} e_2$.
d) $\forall e_1 \in E_{T_c,fw}$ and $\forall e_2 \in E_{T_c,bw}$, if $j_1 \leq i_2$, then $e_1 \prec_{T_c,bw+fw} e_2$.

The combination of the three partial orders defined above enforces a linear order $\prec_{T_c,E_c}$ on E_c.

Given this linear order $\prec_{T_c,E_c}$, we can order all edges in G_c and construct an edge sequence to form a DFS code of a CRM, denoted as $code(c, T_c)$. An edge of the DFS code of a CRM is represented similar to the CRM edge definition and uses a 5-tuple representation $(i, j, \boldsymbol{l_i}, \boldsymbol{l_{i,j}}, \boldsymbol{l_j})$, where i and j are the DFS subscripts (i.e., the discovery time) of the vertices, and $\boldsymbol{l_i}$, $\boldsymbol{l_j}$, and $\boldsymbol{l_{i,j}}$ are the label vectors of the vertices and edge, respectively. The kth entry in each vector $\boldsymbol{l_i}$, $\boldsymbol{l_j}$, and $\boldsymbol{l_{i,j}}$ contains the labels of vertices and edges of motif M_k.

For example, Figure 4 presents two CRMs and their corresponding DFS codes. The DFS codes are listed below to show the sequence of edges and their differences (i.e., the edge labels):

CRM C_1 - Figure 4 (a)	CRM C_2 - Figure 4 (c)
$\langle(0, 1, \langle \mathtt{a}, \mathtt{a}, \mathtt{b}\rangle, \langle \mathtt{X}, \mathtt{Y}, \mathtt{X}\rangle, \langle \mathtt{b}, \mathtt{c}, \mathtt{c}\rangle)$,	$\langle(0, 1, \langle \mathtt{a}, \mathtt{a}, \mathtt{b}\rangle, \langle \mathtt{X}, \mathtt{Y}, \mathtt{X}\rangle, \langle \mathtt{b}, \mathtt{c}, \mathtt{c}\rangle)$,
$(1, 2, \langle \mathtt{b}, \mathtt{c}, \mathtt{c}\rangle, \langle \mathtt{Y}, \omega, \mathtt{Y}\rangle, \langle \mathtt{c}, \mathtt{c}, \mathtt{f}\rangle)$,	$(1, 2, \langle \mathtt{b}, \mathtt{c}, \mathtt{c}\rangle, \langle \mathtt{Y}, \mathtt{Z}, \omega\rangle, \langle \mathtt{c}, \mathtt{c}, \mathtt{f}\rangle)$,
$(2, 0, \langle \mathtt{c}, \mathtt{c}, \mathtt{f}\rangle, \langle \mathtt{Z}, \omega, \omega\rangle, \langle \mathtt{a}, \mathtt{a}, \mathtt{b}\rangle)$,	$(2, 0, \langle \mathtt{c}, \mathtt{c}, \mathtt{f}\rangle, \langle \omega, \omega, \mathtt{Z}\rangle, \langle \mathtt{a}, \mathtt{a}, \mathtt{b}\rangle)$,
$(2, 3, \langle \mathtt{c}, \mathtt{c}, \mathtt{f}\rangle, \langle \omega, \mathtt{Z}, \mathtt{Z}\rangle, \langle \omega, \mathtt{d}, \mathtt{g}\rangle)\rangle$	$(2, 3, \langle \mathtt{c}, \mathtt{c}, \mathtt{f}\rangle, \langle \omega, \omega, \mathtt{Z}\rangle, \langle \omega, \mathtt{d}, \mathtt{g}\rangle)\rangle$

Note that the kth entry of a vertex and edge label vector of a DFS code is filled with ω if the corresponding vertex or edge is not present in motif M_k. DFS code's Neighborhood restriction property defined in [Yan and Han 2002] still holds for DFS code of a CRM.

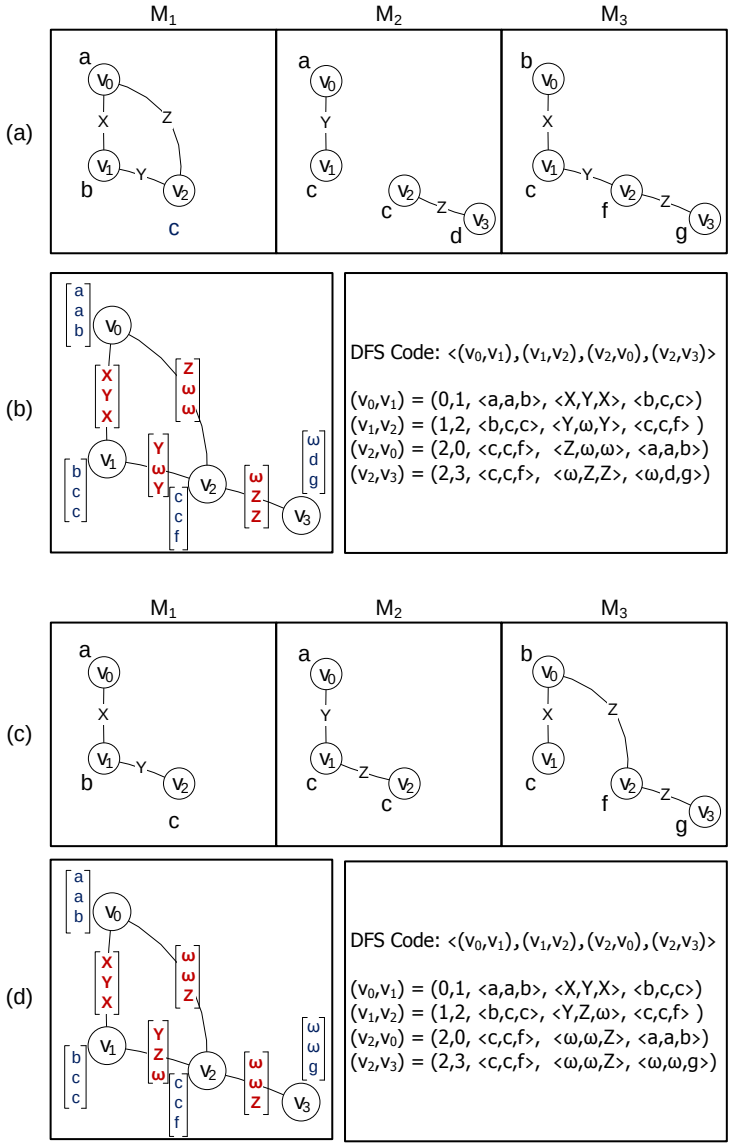


Fig. 4. DFS code for two CRMs represented as a sequence of edges (v_0, v_1), (v_1, v_2), (v_2, v_0), and (v_2, v_3). (a) Presents CRM C_1 consisting of 3 motifs, 4 vertices, and 4 edges. (b) Presents G_{C_1} and the corresponding DFS code for CRM C_1. (c) Presents CRM C_2 consisting of 3 motifs, 4 vertices, and 4 edges. (d) Presents G_{C_2} and the corresponding DFS code for CRM C_2.

5.3.2. DFS Lexicographic Ordering. To establish a canonical labeling system for a CRM, CRMminer defines the DFS lexicographical ordering based on the CRM's DFS code definition. The linear ordering is defined as follows. Let two DFS codes of a CRM consisting of m motifs be $\alpha = code(c_\alpha, T_\alpha) = \langle e^0_\alpha, e^1_\alpha, \ldots, e^p_\alpha \rangle$ and $\delta = code(c_\delta, T_\delta) = \langle e^0_\delta, e^1_\delta, \ldots, e^q_\delta \rangle$, where p and q are the number of edges in α and δ, and $0 \leq p, q$. Let the forward and backward edge set for T_α and T_δ be $E_{\alpha,fw}$, $E_{\alpha,bw}$, $E_{\delta,fw}$, and $E_{\delta,bw}$, respec-

tively. Also, let $e^x_\alpha = (i^x_\alpha, j^x_\alpha, \boldsymbol{l_{i^x_\alpha}}, \boldsymbol{l_{i^x_\alpha, j^x_\alpha}}, \boldsymbol{l_{j^x_\alpha}})$ and $e^y_\delta = (i^y_\delta, j^y_\delta, \boldsymbol{l_{i^y_\delta}}, \boldsymbol{l_{i^y_\delta, j^y_\delta}}, \boldsymbol{l_{j^y_\delta}})$ be two edges, one in each of the α and δ DFS codes. Now, $e^x_\alpha < e^y_\delta$, if any of the following is true:

i) $e^x_\alpha \in E_{\alpha,bw}$ and $e^y_\delta \in E_{\delta,fw}$, or
ii) $e^x_\alpha \in E_{\alpha,bw}$, $e^y_\delta \in E_{\delta,bw}$, and $j^x_\alpha < j^y_\delta$, or
iii) $e^x_\alpha \in E_{\alpha,bw}$, $e^y_\delta \in E_{\delta,bw}$, $j^x_\alpha = j^y_\delta$ and $\boldsymbol{l_{i^x_\alpha, j^x_\alpha}} < \boldsymbol{l_{i^y_\delta, j^y_\delta}}$, or
iv) $e^x_\alpha \in E_{\alpha,fw}$, $e^y_\delta \in E_{\delta,fw}$, and $i^y_\delta < i^x_\alpha$, or
v) $e^x_\alpha \in E_{\alpha,fw}$, $e^y_\delta \in E_{\delta,fw}$, $i^x_\alpha = i^y_\delta$ and $\boldsymbol{l_{i^x_\alpha}} < \boldsymbol{l_{i^y_\delta}}$, or
vi) $e^x_\alpha \in E_{\alpha,fw}$, $e^y_\delta \in E_{\delta,fw}$, $i^x_\alpha = i^y_\delta$, $\boldsymbol{l_{i^x_\alpha}} = \boldsymbol{l_{i^y_\delta}}$ and $\boldsymbol{l_{i^x_\alpha, j^x_\alpha}} < \boldsymbol{l_{i^y_\delta, j^y_\delta}}$, or
vii) $e^x_\alpha \in E_{\alpha,fw}$, $e^y_\delta \in E_{\delta,fw}$, $i^x_\alpha = i^y_\delta$, $\boldsymbol{l_{i^x_\alpha}} = \boldsymbol{l_{i^y_\delta}}$, $\boldsymbol{l_{i^x_\alpha, j^x_\alpha}} = \boldsymbol{l_{i^y_\delta, j^y_\delta}}$, and $\boldsymbol{l_{j^x_\alpha}} < \boldsymbol{l_{j^y_\delta}}$.

Based on the above definitions, we derive the following conditions to compare two DFS codes of a CRM. We define $\alpha \leq \delta$, iff any of the following conditions is true:

a) $F_\omega(\alpha) \leq F_\omega(\delta)$, where $F_\omega(x)$ is the number of edges $(\boldsymbol{l_{i^t_x, j^t_x}})$ that are set to ω, or
b) $\exists t, 0 \leq t \leq min(p, q)$, $e^k_\alpha = e^k_\delta$ for $k < t$, and $e^t_\alpha < e^t_\delta$, or
c) $e^k_\alpha = e^k_\delta$ for $0 \leq k \leq p$ and $p \leq q$.

Note that the DFS lexicographical ordering ranks edges with label vectors containing no ω labels higher than the edges which does. To define the relation between ω and valid vertex/edge label, the value of ω is set to a lexicographically higher value than the maximum edge and vertex label. This is important as we show later in Section 5.3.5. In order to provide a detailed example of the DFS lexicographical ordering, let us compare the DFS codes of CRMs C_1 and C_2 presented in Figure 4 following the rules presented above. For both the DFS codes, the first edge (v_0, v_1) is the same. In case of the second edge (v_1, v_2), both the DFS codes contain the same vertex label vectors. However, the edge label vectors are different and the edge label vector $\langle Y, Z, \omega \rangle$ of DFS code 2 is smaller than $\langle Y, \omega, Y \rangle$ DFS code 1. Thus, DFS code 2 is lexicographically smaller than DFS code 1.

5.3.3. Minimum DFS Code. A CRM can be represented by different DFS trees resulting in different DFS codes. To define a canonical label for a CRM, we select the minimum DFS code according to the DFS lexicographic order, represented by $min\_code(c)$. Similar to simple graph isomorphism, given two CRMs c and c', c is isomorphic to c' if and only if $min\_code(c) = min\_code(c')$. Thus, by searching frequent minimum DFS codes, we can identify the corresponding frequent CRMs.

5.3.4. Pattern lattice growth. The difference between a simple graph pattern and a CRM is the vertex/edge label representation. One contains a single label and the other contains a label vector (i.e., sequence of labels including an empty label ω). The growth of a simple graph by one edge results in a number of simple graphs based on the number of unique labels of the frequent edges on the rightmost path. However, a CRM extended by an edge can generate large number of CRMs, since these are formed based on the combination of the label vectors of the frequent edges on the rightmost path.

In Figure 5, we illustrated one edge growth of a simple graph and a CRM according to rightmost extension. Both the simple graph and the CRM are expanded by adding a frequent edge (v_2, v_3). In case of the simple graph (Figure 5(a)), the original DFS code consisted sequence of two edges (v_0, v_1) and (v_1, v_2) represented as $(0, 1, a, X, b)$ and $(1, 2, b, Y, c)$. After the one edge extension, the new edge $(2, 3, c, Z, d)$ connected the new vertex v_3 to the rightmost vertex v_2. When the CRM is considered (Figure 5(c)), the original DFS code consisted the same sequence of edges $\{(v_0, v_1), (v_1, v_2)\}$ represented as: $(0, 1, \langle a, a, b \rangle, \langle X, Y, X \rangle, \langle b, c, c \rangle)$, $(1, 2, \langle b, c, c \rangle, \langle Y, \omega, Y \rangle, \langle c, \omega, f \rangle)$. Based on the combination of the label vectors of vertices v_2 and v_3, and edge (v_2, v_3), we have the

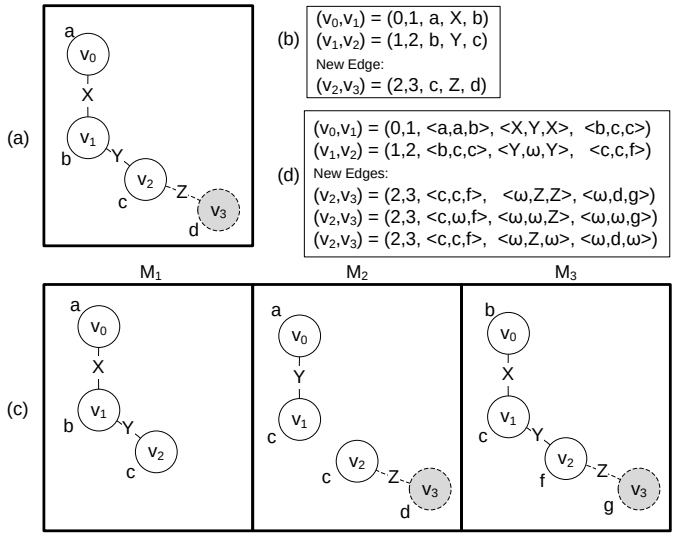


Fig. 5. Adding an edge according to rightmost extension rules. (a) Extending a simple graph, (b) DFS code of the simple graph, (c) extending a CRM, and (d) DFS code of the CRM.

following options for the (v_2, v_3) edge: $(2, 3, \langle c, c, f\rangle, \langle \omega, Z, Z\rangle, \langle \omega, d, g\rangle)$, $(2, 3, \langle c, \omega, f\rangle, \langle \omega, \omega, Z\rangle, \langle \omega, \omega, g\rangle)$, and $(2, 3, \langle c, c, f\rangle, \langle \omega, Z, \omega\rangle, \langle \omega, d, \omega\rangle)$. Note that we allow a CRM to grow by an edge that may not be present or frequent in all motifs of that CRM to ensure complete set of the results.

To efficiently determine the frequent candidate edges during the rightmost extension, we apply the sequential pattern mining technique [Pei et al. 2001a]. Even though there can be significantly more number of child CRMs from one edge extension of a CRM than a simple graph, the extended lexicographic ordering is able to order all candidates and enable us to perform pre-order search on the pattern lattice. Since traversal of the pattern lattice of a CRM remains similar to a simple graph, it allows CRMminer to prune the pattern with non-minimum DFS codes and their descendants similar to gSpan without impacting the completeness of the results.

5.3.5. Algorithm Completeness. To eliminate redundancy during the CRM expansion process, we use minimum DFS code as the canonical label and construct the pattern lattice to ensure that every node (i.e., a CRM) is connected to a unique parent and grown via single edge addition. This process ensures that each potential CRM is only explored once.

To ensure that all discovered CRMs contain at least one anchor, we show how we can identify an anchor of a valid CRM. Given a valid CRM c and its canonical label (i.e., the minimum DFS code) $min\_code(c) = \langle e_1, e_2, \ldots, e_k\rangle$, where e_i is an edge in the lexicographically ordered edge sequence. We claim that the first edge (e_1) of the canonical label of a CRM (c) is an anchor of that CRM. To prove this claim, assume e_1 is not an anchor and e_x is an anchor, where $1 < x \leq k$. Given the graph in CRM c, we can construct a DFS code for c that starts with e_x as the first edge. Assume, the new DFS code is represented as $code(c) = \langle e_x, e_{p_1}, \ldots, e_{p_{k-1}}\rangle$, where $\langle e_{p_1}, \ldots, e_{p_{k-1}}\rangle$ is an edge sequence containing a permutation of the edges $\{e_1, \ldots, e_k\} \setminus \{e_x\}$. Since e_1 is not an anchor, it contains some ω labels. Based on the DFS lexicographic ordering and e_x being an anchor, $e_x < e_1$. Hence, we can state that $\langle e_x, e_{p_1}, \ldots, e_{p_{k-1}}\rangle < \langle e_1, e_2, \ldots, e_k\rangle$. This is a contradiction, since $\langle e_1, e_2, \ldots, e_k\rangle$ is the minimum DFS code of c. Thus, e_1 is

an anchor of CRM c. Given that the first edge is an anchor, then CRMminer will generate that CRM by starting from the anchor and then following its right-most extension rule to add the rest of the edges one by one.

5.4. Search space pruning

One of the challenges that any graph mining algorithm needs to handle is the exponential growth of the search space during enumeration. Traditionally, user specified constraints are used to prune the search space. To ensure discovery of the complete set of patterns, the pruning constraints need to have the anti-monotonicity property. For CRMminer, we use both support measure and minimum overlap constraints to prune the search space.

5.4.1. Minimum Support (ϕ). To efficiently search patterns in a single large graph using a minimum support constraint, the support measure needs to guarantee the anti-monotonicity property. Bringmann [Bringmann and Nijssen 2008] presented the minimum image based support measure to prune the search space in a single large graph. Given a pattern $p = (V_p, E_p, L_p)$ and a single large graph $G = (V_G, E_G, L_G)$, this measure identifies the vertex in p which is mapped to the least number of unique vertices in G and uses this number as the frequency of p in G. To formally define the support measure, let each subgraph g of G that is isomorphic to p be defined as an *occurrence* of p in G. For each occurrence g, there is a function $\varphi : V_p \rightarrow V_G$ that maps the nodes of p to the nodes in G such that (i) $\forall v \in V_p \Rightarrow L_p(v) = L_G(\varphi(v))$ and (ii) $\forall (u, v) \in E_p \Rightarrow (\varphi(u), \varphi(v)) \in E_G$. The minimum image based support of a pattern p in G is defined as:

$$\sigma(p, G) = \min_{v \in V_p} | \{\varphi_i(v) \text{: } \varphi_i \text{ is an occurrence of } p \text{ in } G\} |. \quad (1)$$

This minimum image based support is anti-monotonic [Bringmann and Nijssen 2008].

We adopted the minimum image based support measure to calculate the minimum support of a CRM in a dynamic network. As defined in Section 2, a dynamic network $\mathcal{N}$ can be represented as a single large graph where the nodes $\mathcal{N}$ are considered as the vertices of the large graph. Hence, it is possible to calculate the least number of unique vertices of the dynamic network that are mapped to a particular vertex of a CRM. Given a CRM $c = \{N_c, \langle M_1, M_2, \ldots, M_m \rangle\}$ in a dynamic network $\mathcal{N} = \{V_{\mathcal{N}}, \langle G_0, G_1, \ldots, G_T \rangle\}$ where $m \leq T$, the minimum image based support of c is defined as:

$$\sigma(c, \mathcal{N}) = \min_{v \in V_c} | \{\varphi_i(v) \text{: } \varphi_i \text{ is an occurrence of } c \text{ in } \mathcal{N}\} |. \quad (2)$$

Similar to the support measure of a pattern in a single large graph, by selecting the support of the vertex in c that has the least number of unique mapping in $\mathcal{N}$, we maintain the anti-monotonicity property.

Recall from Section 5.3.4 that the frequent candidate edges are identified using frequent sequence mining. Since we use the minimum image based support measure, the frequent edges detected by the sequence mining tool may not have sufficient support when calculated using such measure. Thus, we compute the minimum image based support for all candidate edges to only consider the edges that ensures the CRM extension to have sufficient minimum image based support.

5.4.2. Minimum Overlap (β). Each motif of a CRM needs to contain at least a minimum percentage of the nodes from all the nodes of the CRM. This minimum node

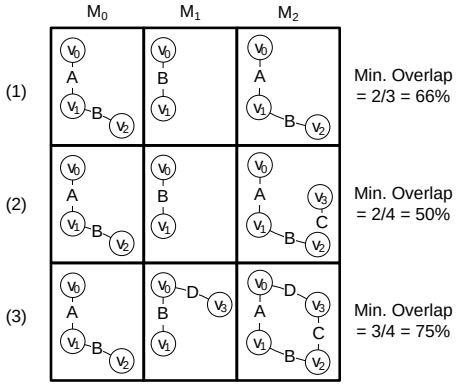


Fig. 6. Example of a CRM growth when the minimum overlap constraint is not anti-monotonic. Assume the minimum overlap constraint is 60%.

overlap threshold (defined as β in Section 4) controls the degree of change that is allowed between the sets of nodes in each motif of a CRM. Given a CRM $c = \{N_c, \langle M_1, M_2, \ldots, M_m \rangle\}$ containing m motifs, the *minimum overlap* of a CRM is defined as:

$$\rho(c) = \frac{\min\limits_{1 \leq i \leq m} \{|V_{M_i}|\}}{|N_c|}, \tag{3}$$

where V_{M_i} is the set of nodes in motif M_i. Even though the minimum overlap constraint is a reasonable approach to ensure that the motifs that make the CRM are coherent, it is not anti-monotonic [Zhu et al. 2007]. Thus, to generate a complete set of CRMs that meet user specified thresholds of support and overlap, we cannot prune CRMs that do not satisfy this constraint as CRMs derived from it can satisfy the constraint.

For example, let us assume the minimum overlap threshold is 60% in Figure 6. In step (1), motif M_1 contains 2 out of 3 nodes of the CRM (i.e., the minimum). Therefore, the minimum overlap at step (1) $(2/3 > 60\%)$ is valid. We added vertex v_3 by including edge (v_2, V_3) to the CRM at step (2). This dropped the minimum overlap $(2/4)$ below the threshold. However, in step (3), inclusion of edge (v_3, v_0) adds vertex v_3 to motif M_1. This increases the minimum overlap to be $3/4$ and makes the CRM valid again. Hence, we need to enumerate all CRMs that meet the support threshold and then search the output space for CRMs that meet the minimum overlap requirement.

To improve the performance, we developed an approximate version of our algorithm, named CRMminer$_{\mathbf{x}}$, that discovers a subset of the valid CRMs (i.e., meet the constraints of Definition 4.1) by pruning the pattern lattice using the minimum overlap threshold. We first check whether a CRM meets the overlap threshold. If it does, we continue the enumeration process. If it does not, then we check whether any of the patterns at the previous level of the pattern lattice from which the current pattern was derived, referred as parent CRMs, meet the overlap threshold. If at least one does, we continue enumeration. If none of the parent CRMs meet the overlap threshold, we prune that CRM.

To approximately calculate the minimum overlap of the parent CRMs, we do not generate all possible parent patterns. The parent is defined to contain one less node than the current CRM; thus, we remove a node $v \in N_c$. In such case, the parent pattern will contain $(|N_c| - 1)$ nodes and each motif M_i may contain $(|V_{M_i}| - 1)$ nodes if $v \in V_{M_i}$ or $(|V_{M_i}|)$ nodes if $v \notin V_{M_i}$. For at least one parent CRM to meet the overlap threshold,

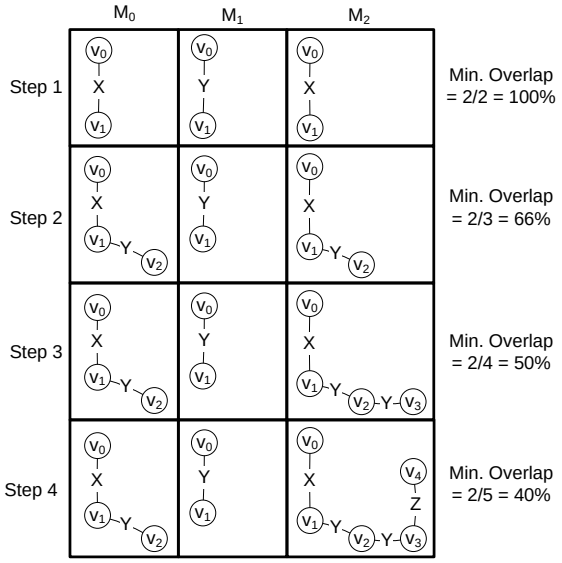


Fig. 7. Minimum overlap calculation based on (4) is used for search space pruning during the CRM enumeration process. Assuming $\beta = 60\%$, this CRM enumeration terminates at step 4.

we consider the best case when $v \notin V_{M_i}$. Therefore, the minimum overlap threshold of a parent CRM c_p of the CRM $c = \{N_c, \langle M_1, M_2, \ldots, M_m \rangle\}$ is defined as:

$$\rho(c_p) = \frac{\min\limits_{1 \leq i \leq m} \{|V_{M_i}|\}}{|N_c| - 1}. \tag{4}$$

In Figure 7, we illustrate the minimum overlap calculation during search space pruning. Assume the user specified $\beta = 60\%$. We start with the anchor at step 1 when the minimum overlap threshold is 100%. Next the edge (v_1, v_2) is added for motif M_0 and M_2 and the minimum overlap based on motif M_1 is $(2/3) = 66\%$. Since the β threshold is met, we continue the enumeration process. At step 3, an edge (v_2, v_3) is added to motif M_2. Since motif M_1 contains the lowest number of nodes, the minimum overlap is $(2/4) = 50\%$, which does not meet the β threshold. At this point, we check whether any of the parent CRM meets the β threshold and the minimum threshold for it's parent is $(2/(4-1)) = 66\% > \beta$. Thus, we continue the enumeration by adding an edge (v_3, v_4) to motif M_2 at step 4. The minimum overlap is $(2/5) = 40\%$ and it's parent overlap threshold is $(2/(5-1)) = 50\%$. Both thresholds are lower than β, hence we stop enumerating this CRM any further.

6. EXPERIMENTAL DESIGN

All experiments are conducted on a 64-bit Linux desktop with 8-core Intel Core i7-3770 processor at 3.40GHz and 16GB of RAM.

6.1. Datasets

We have used two different types of datasets to evaluate CRMminer. The DBLP co-authorship network is a real world dynamic network that captures yearly co-authorship relations. The bioprocess network (GT) and the sales network (Sales) datasets are based on multivariate time-series data. To characterize the relations among different variables and understand changes over time, we represent the time-

Table I. Dynamic Network Datasets.

Dataset	#Vertices	#Edeges 66	Span	Avg. #Edeges	Avg. Degree
DBLP	1,057,524	4,841,084	55	88,020	0.08
GT	3458	83,454	47	1776	0.51
Sales	2697	138,044	66	2092	0.78

#Vertices denotes the total number of vertices in the dynamic network. #Edeges denotes the total number of edges in the dynamic network. Span denotes the total number of snapshots in the dynamic network. Avg. #Edeges denotes the average number of edges per snapshot in the dynamic network. Avg. Degree denotes the avg. number of edges per node in a snapshot.

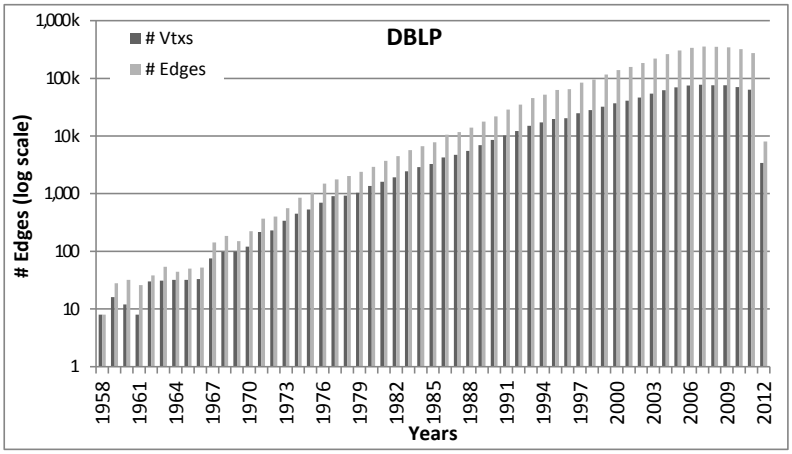


Fig. 8. The edge distribution of the DBLP co-authorship network.

series data as a dynamic network. The CRMs discovered from these networks can be used to characterize the overall network, as shown in Section 7.2.3.

6.1.1. Co-Authorship Network (DBLP). This is a co-authorship network that models the yearly co-authorship relations from 1958 to 2012 based on the DBLP Computer Science Bibliography Data [Ley 2008]. The snapshots of the dynamic network that we created corresponds to the co-authorship network of each year, leading to a dynamic network consisting of $2012 - 1958 = 55$ snapshots. The nodes model the authors of the publications from various conferences, journals, books, lecture series, etc. and the undirected edges between the authors model the collaboration between two scientists at a certain year. If an author A co-authors a publication with an author B in a given year, the edge (A, B) is added to that year's co-authorship graph. Multiple publications by the same authors in a year are counted as a single relation.

The edge distribution among the snapshots is skewed. In Figure 8, we show the edge distribution of the DBLP network. To determine co-authorship relations between authors with significant contribution, we removed authors who published less than 5 different years. We also removed all the relations of an author if he/she had co-authored with more than 50 other authors in a single year. To assign edge labels, we used the CLUTO software[1] to cluster the publication titles into 50 groups. These title groups can be thought as the research areas or publication topics that the authors focused on their collaboration.

[1]http://www.cs.umn.edu/~cluto

Table II. Bioprocess Network Dataset

Vertex label	Variable(s) Description
ASX	Air sparge rate, Air sparge total
BT	Base totalized
CO2	CO_2 sparge rate, CO_2 sparge total
DOX	Dissolved oxygen controller output, Dissolved oxygen primary
FRO	Flowrate overlay
O2X	O_2 sparge rate, O_2 sparge total
PHX	pH Controller output, pH Online
ST	Sparge total
WLC	Weight load cell

The vertex labels of the 14 parameters used to construct the Bioprocess Network Dataset (GT) dataset. The labels ASX, CO2, DOX, O2X, and PHX represent two parameters each.

6.1.2. Bioprocess Network (GT). This is a cell culture bioprocess data [Le et al. 2012] captured from the production bioreactors at Genentech's manufacturing facility. This multivariate time-series data tracks the dynamics of various process parameters at every minute over 11 days period for a 247 production runs. In Table II, we show the parameters that are used to construct the GT network. Note that some of the related variables are assigned the same vertex label. To represent this data as a dynamic network, we computed correlation matrices for 14 of the process parameters using a sliding window of 12 hours interval with 50% overlap. This process resulted in 47 correlation matrices per production run. To construct a network snapshot based on a correlation matrix, we use each parameter as a vertex and two parameters/vertices are connected with an edge labeled as positive/negative if their correlation is above $+0.9$ or below -0.8 threshold. These threshold values are chosen to select equal number of positive and negative labeled edges. This way we construct 47 network snapshots where each snapshot contains a 3458 vertices (14 parameter * 247 runs) and the edges between them.

6.1.3. Store Transaction Network (Sales). This dynamic network is constructed using Dominicks Finer Foods store sales data collected from the James M. Kilts Center, University of Chicago Booth School of Business[2]. The data captures weekly store-level sales from 93 stores collected over a period of more than seven years (400 weeks). We considered the sales data as multivariate time-series data for each store where 29 variables are the different categories of the product sold. In Table III, we show the different categories of the product considered in the Sales network. Note that some of the related variables are assigned the same vertex label. To represent this data as a dynamic network, we considered the total number of items sold per category in a week at a store and computed correlation matrices between 29 categories using a sliding window of 12 weeks interval with 50% overlap. This process resulted in 66 correlation matrices per store. To construct a network, we use each category as a vertex and two vertices are connected with an edge labeled as positive or negative if their correlation is above $+0.85$ or below -0.4 respectively. These threshold values are chosen to select equal number of positive and negative labeled edges. This way we construct 66 network snapshots where each snapshot contains a 2697 vertices (29 parameters * 93 stores) and the edges between them.

6.2. Metrics

In order to asses the scalability and performance of the algorithm, we collect two sets of results for the discovered CRMs. The first set of results is collected by running

[2]http://research.chicagobooth.edu/kilts/marketing-databases/dominicks/dataset

Table III. Sales Dataset

Vertex label	Variable(s) Description
ANA	Analgesics
BER	Beer
CEX	Cereals, Oatmeal
CHE	Cheeses
CIG	Cigarettes
CRX	Cookies, Crackers, Snack Crackers
CSO	Canned Soup
DID	Dish Detergents
DTX	Laundry Detergents, Fabric Softeners
FEC	Front-end-candies
FRX	Frozen Dinners, Entree, Juices
GRO	Grooming Products
JUX	Bottled Juices, Refrigerated Juices
PTW	Paper Towels
SDR	Soft Drinks
SPX	Bath Soap, Soaps, Shampoos
TEX	Toothbrushes, Toothpastes
TNA	Canned Tuna
TTI	Bathroom Tissues

The vertex labels of the 29 parameters used to construct the Sales dataset. The labels CRX, FRX, and SPX represent three parameters each. The labels CEX, DTX, JUX, and TEX represent two parameters each.

CRMminer, that does not perform any overlap based pruning during CRM enumeration. This process generates the complete set of CRMs (T_{CRM}) for the specified support threshold and then applies the overlap threshold to identify the valid CRMs (Q_{CRM}) from the output space. The second set of results are collected by running CRMminer$_{\text{x}}$, that applies the overlap threshold to prune the search space during enumeration. This process uses Equation (4) to perform the overlap threshold check for search space pruning to collect all CRMs (T_{CRM}) and then performs a final check using Equation (5) to select the valid CRMs (Q_{CRM}). As discussed in Section 5.4.2, overlap based pruning does not guarantee complete set of results. Thus, the set of CRMs in T_{CRM} and Q_{CRM} collected using the CRMminer$_{\text{x}}$ are subsets of T_{CRM} and Q_{CRM} collected by using the CRMminer correspondingly.

7. RESULTS

The evaluation consists of two parts. The first focuses on assessing the performance of the algorithm that we developed for finding CRMs and to assess how the different parameters associated with the definition of CRMs impacts the performance. The second focuses on assessing the information that can be extracted from the discovered CRMs by analyzing some of the patterns of coevolving relational motifs that were identified in the three datasets.

7.1. Performance Results

7.1.1. Minimum Support & Overlap. Table IV and Figure 9 show the performance of the CRM mining algorithms and the size distribution of the discovered CRMs, respectively. These results are presented for different values of the minimum support and overlap thresholds.

The following observations can be made from these results. First, the number of discovered CRMs increases as the minimum support decreases and/or the amount of overlap decreases. Both of which were expected. Moreover, as the minimum support and/or overlap decrease, the size of the discovered CRMs increases. Second, the amount of

Table IV. Minimum Support (ϕ) & Overlap (β) study.

Data	ϕ	β	#A	CRMminer			CRMminer$_x$		
				#T_{CRM}	#Q_{CRM}	Time	#T_{CRM}	#Q_{CRM}	Time
DBLP	100	0.70	9	261	84	27.00	261	84	26.05
		0.60	”	”	84	”	261	84	26.28
		0.50	”	”	261	”	261	261	26.49
		0.40	”	”	261	”	261	261	27.22
	90	0.70	14	548	126	61.99	496	126	49.09
		0.60	”	”	133	”	496	133	47.48
		0.50	”	”	496	”	548	496	62.06
		0.40	”	”	548	”	548	548	54.09
	80	0.70	16	1,724	203	2,298.84	1,044	203	109.78
		0.60	”	”	225	”	1,044	225	120.15
		0.50	”	”	1,044	”	1,399	1,044	239.89
		0.40	”	”	1,492	”	1,558	1,491	694.31
GT	90	0.70	7	6,255	1,370	7.10	4,553	1,370	5.92
		0.60	”	”	3,724	”	4,730	3,410	5.92
		0.50	”	”	5,173	”	6,027	5,173	7.04
		0.40	”	”	6,255	”	6,255	6,255	7.18
	80	0.70	7	240,828	145,633	200.44	216,233	144,243	182.92
		0.60	”	”	205,674	”	224,188	200,439	184.44
		0.50	”	”	231,222	”	240,376	231,222	198.87
		0.40	”	”	240,828	”	240,828	240,828	199.59
	70	0.70	9	973,116	626,877	730.21	804,228	611,206	611.04
		0.60	”	”	796,897	”	871,176	778,893	685.46
		0.50	”	”	922,324	”	958,742	919,640	712.06
		0.40	”	”	973,116	”	973,116	973,116	714.63
Sales	45	0.70	65	11,917	134	18.70	1,472	134	3.52
		0.60	”	”	134	”	1,472	134	3.49
		0.50	”	”	1,472	”	5,785	1,472	9.75
		0.40	”	”	10,480	”	11,908	10,480	18.84
	40	0.70	90	183,437	551	207.75	7,422	551	14.03
		0.60	”	”	1,027	”	7,741	1,027	14.39
		0.50	”	”	8,582	”	37,109	8,582	52.46
		0.40	”	”	99,174	”	156,320	99,174	182.53
	35	0.70	109	24,509,851	47,535	11,904.75	263,475	38,429	299.80
		0.60	”	”	552,297	”	1,205,965	351,611	904.21
		0.50	”	”	3,701,651	”	4,950,385	2,617,409	3,040.26
		0.40	”	”	13,780,709	”	18,563,088	13,427,964	8,327.78

ϕ denotes the minimum support. #A denotes the number of discovered anchors. #T_{CRM} denotes the total number of discovered CRMs. #Q_{CRM} denotes the number of CRMs that meet the β threshold out of #T_{CRM}. Time denotes the amount of time spent in seconds expanding the anchors to discover Q_{CRM}. For all datasets, k_{min} is 4, k_{max} is 10, and m_{min} is 4.

time required by CRMminer$_x$ is lower than that required by CRMminer. Depending on the experiment, it is $1-40$ times faster than CRMminer with an average speedup of 5.8. The performance gap is higher for large values of overlap and progressively shrinks as the overlap decreases. This is expected, as CRMminer$_x$'s overlap-based pruning becomes less effective for low overlap values. Third, the number of CRMs missed by the approximate nature of CRMminer$_x$ is either none or relatively small. This indicates that CRMminer$_x$ is a viable algorithm for CRM discovery as it is both faster and also quite effective in finding most valid CRMs.

Finally, comparing how the two algorithms scale with the size of the output space (i.e., the number of valid discovered CRMs), we see that for most datasets, they ei-

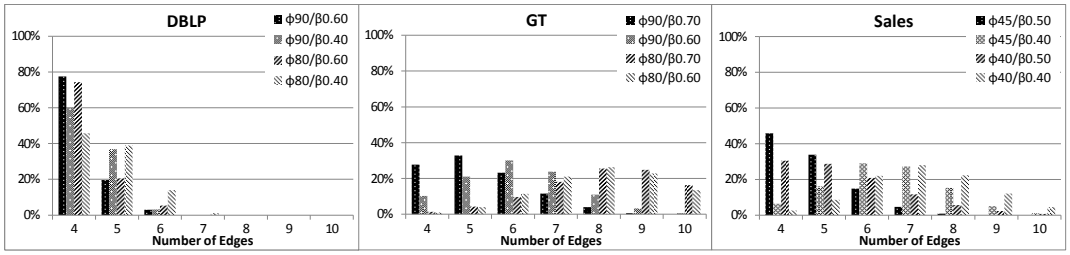
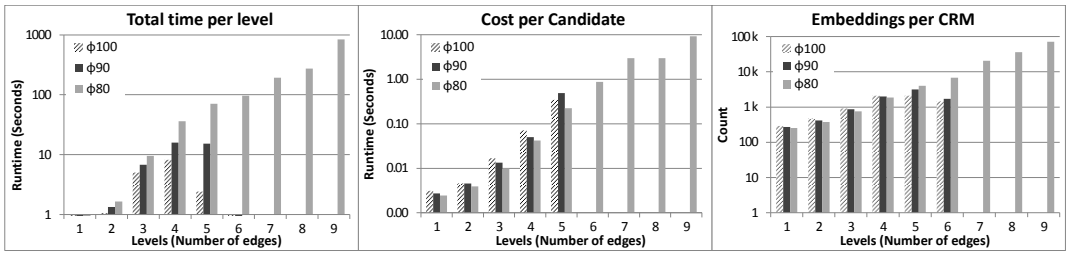


Fig. 9. CRMs size distribution of different datasets for different minimum support and overlap thresholds. For all datasets, $k_{min} = 4$, $k_{max} = 10$, and $m_{min} = 4$.



(a) (b) (c)

Fig. 10. CRM enumeration details at each level of the extension for the $DBLP$ dataset. (a) Shows the total time spent at each level. (b) Shows the average time needed to generate a candidate CRM at each expansion level. The cost is calculated as the sum of the average time to locate a frequent edge, the average time to perform minimum DFS code check, and the average time to locate the embeddings of the candidate CRM. (c) shows the average number of embeddings maintained by each CRM at each level. The results in the Y-axis are in log scale. For all experiments, $\beta = 0.60$ and $m_{min} = 4$.

ther scale linearly or better than linearly. The only exception is the $DBLP$ run of CRMminer for $\phi = 80$. For this experiment, CRMminer took 37 times more time than the $\phi = 90$ experiment and depending on the specific overlap value, it only discovered $1.6-2.7$ times more CRMs. To better understand CRMminer's behavior for this dataset, Figure 10 shows various statistics about the amount of time required by the different phases of the algorithm and the number of embeddings of the discovered CRMs. From these results we can see that the reason for the dramatic increase in runtime is due to the fact that for $\phi = 80$ $DBLP$ contains a large number of candidate CRMs that contain many edges (Figure 10(a)) and also have a large number of embeddings (Figure 10(c)). As a result, CRMminer spends most of its time processing these large embedding lists, leading to its substantial increase in runtime. However, most of these candidate CRMs fail to meet the overlap constraint and this is the reason that CRMminer$_{\text{x}}$ performs better and scales better.

Understanding the missing CRMs. The results presented in Table IV show that CRMminer$_{\text{x}}$ is able to find almost all the valid CRMs for the $DBLP$ and the $Sales$ datasets in less time than CRMminer. For the GT dataset, even though the runtime decreased significantly, only a subset of the valid CRMs were found by CRMminer$_{\text{x}}$. As we have seen in the size distribution characteristic of the CRMs of the GT dataset presented in Figure 9, the CRMminer$_{\text{x}}$ fails to discover all CRMs, since the CRMs that become valid at later stage will get eliminated by the overlap threshold based pruning. The lack of anti-monotonicity property for the overlap threshold based prun-

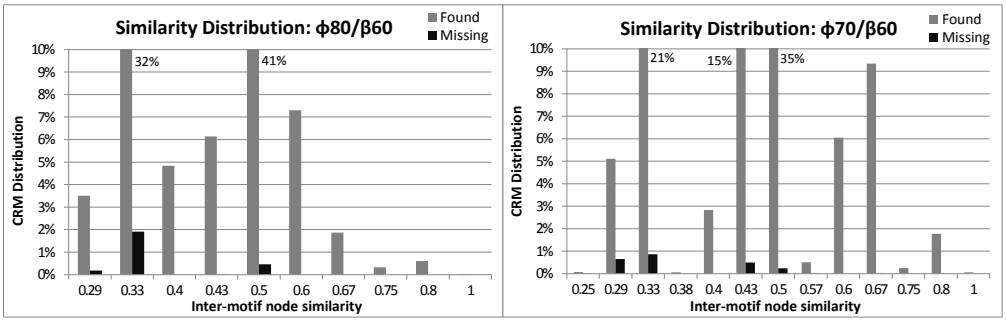


Fig. 11. CRMs distribution based on the minimum inter-motif similarity using the GT dataset.

ing strategy eliminated a large number of potential CRMs at the early stages of the enumeration process.

To understand the characteristics of the missing CRMs, we calculated the *minimum inter-motif similarity* of all CRMs based on the following equation:

$$\psi(c) = \frac{\min\limits_{1<i<j<m} \{|V_{M_i} \cap V_{M_j}|\}}{|N_c|}, \tag{5}$$

where $|V_{M_i} \cap V_{M_j}|$ is the number of common vertices between motif M_i and M_j and $|N_c|$ is the total number of vertices in CRM c.

We analyzed the GT dataset results presented in Table IV for two different experiments ($\phi = 80, \beta = 60$ and $\phi = 70, \beta = 60$). Figure 11 shows the minimum inter-motif similarity distribution of all identified CRMs from two separate experiments using GT datasets. The *found* bars represent the CRMs that were identified by CRMminer$_x$ and the *missing* bars represent the CRMs that were missed due to overlap based pruning during enumeration step. These plots show that the minimum inter-motif similarity among the missing CRMs are low. Hence, these missing CRMs contain motifs that are mostly different from each other in terms of their nodes and resulting in CRMs that may not be capturing interesting relational changes.

7.1.2. Minimum Span. The performance of the algorithm for different values of the minimum span (m_{min}) is shown in Table V. The value of m_{min} represents the minimum number of motifs per CRM. From the reported results in Table V, we observe that

Table V. Minimum Span (m_{min}) study.

		$m_{min} = 3$				$m_{min} = 4$			
Data	ϕ	#A	#T_{CRM}	#Q_{CRM}	Time	#A	#T_{CRM}	#Q_{CRM}	Time
DBLP	120	334	84,950	25,610	12,858.88	9	70	34	5.54
	110	410	116,968	35,582	17,204.75	9	140	55	10.18
	100	502	165,365	50,580	22,577.21	9	263	86	17.49
GT	40	12	1,834,449	1,210,710	381.13	1	19,797	6,490	5.92
	35	15	3,092,486	2,059,379	551.42	2	51,698	16,554	12.47
	30	20	5,150,967	3,392,215	717.02	4	123,853	47,702	23.36
Sales	35	109	425,867	46,832	135.33	11	124	7	0.26
	30	136	3,064,882	441,652	649.85	18	7,642	156	6.25
	25	175	10,840,277	1,379,708	1,817.96	33	518,390	1,308	139.32

m_{min} denotes the minimum number of motifs per CRM, and rest of the column labels are described in Table IV. For all datasets, β is 0.60, k_{min} is 4 and k_{max} is 6.

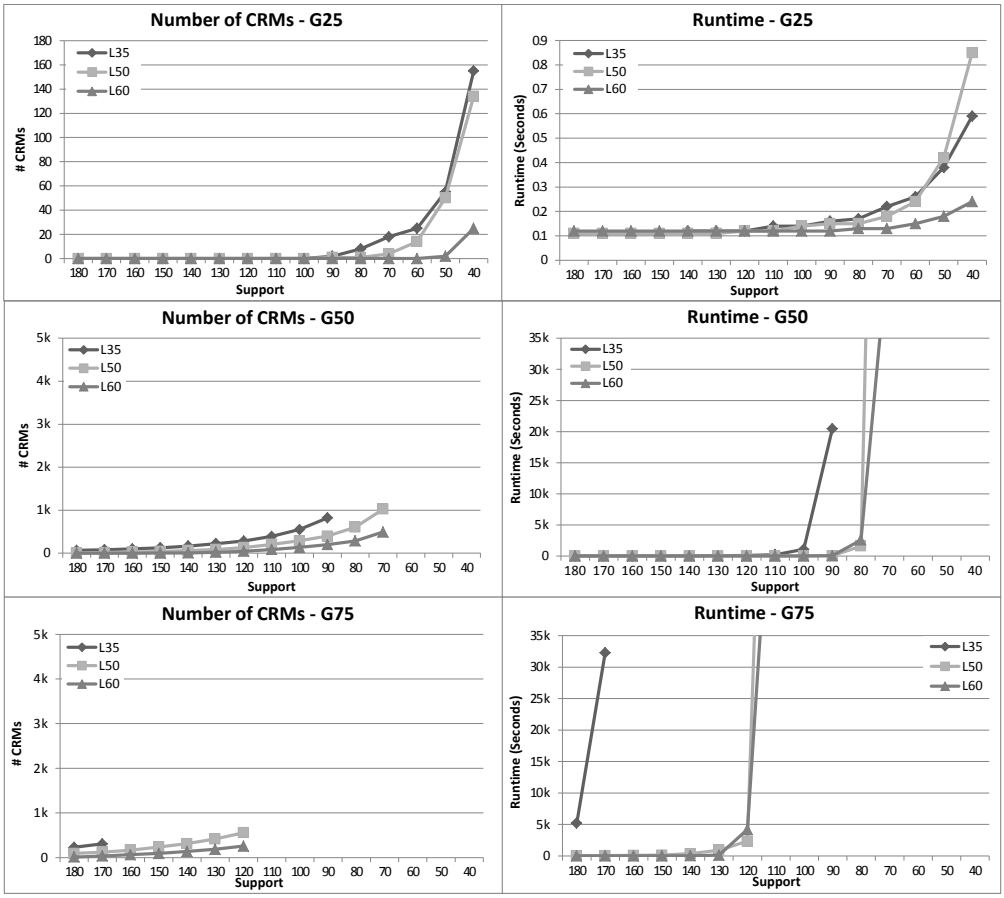


Fig. 12. Performance of CRMminer for different versions of the $DBLP$ dataset. For all experiments, β is 0.60, m_{min} is 4, k_{min} is 3 and k_{max} is 10.

as the value of m_{min} decreases, the number of discovered CRMs and the runtime increases. The value of m_{min} directly impacts the number of anchors and as the number of anchors increases the total number of CRMs increases.

For the $DBLP$ dataset with $\phi = 120$, the number of anchors increases from 9 to 334 for $m_{min} = 4$ to $m_{min} = 3$. As a result, the number of valid CRMs discovered by CRMminer increases from 34 to $25,610$ and the CRMminer runtime increases by 2321 times. We observe similar increase in number of CRMs discovered and runtime for other support thresholds. For the GT dataset, the increase in the number of CRMs and the runtime is less significant than $DBLP$ dataset. As the number of anchors increased from 4 to 20 for $m_{min} = 4$ to $m_{min} = 3$ with $\phi = 30$, the total number of CRMs ($\#Q_{\text{CRM}}$) increased by 72 times and the runtime increased by 31 times. For the $Sales$ dataset, we observe similar increase in both the number of CRMs and the runtime.

7.1.3. Label Diversity & Network Density. The performance of a CRM mining algorithm, as well as any pattern mining algorithm is primarily impacted by the label diversity and the network density of the dataset. To evaluate the performance of our algorithms for different types of dynamic network datasets, we used the $DBLP$ dataset to generate nine different networks varying the number of edge labels and the density of the

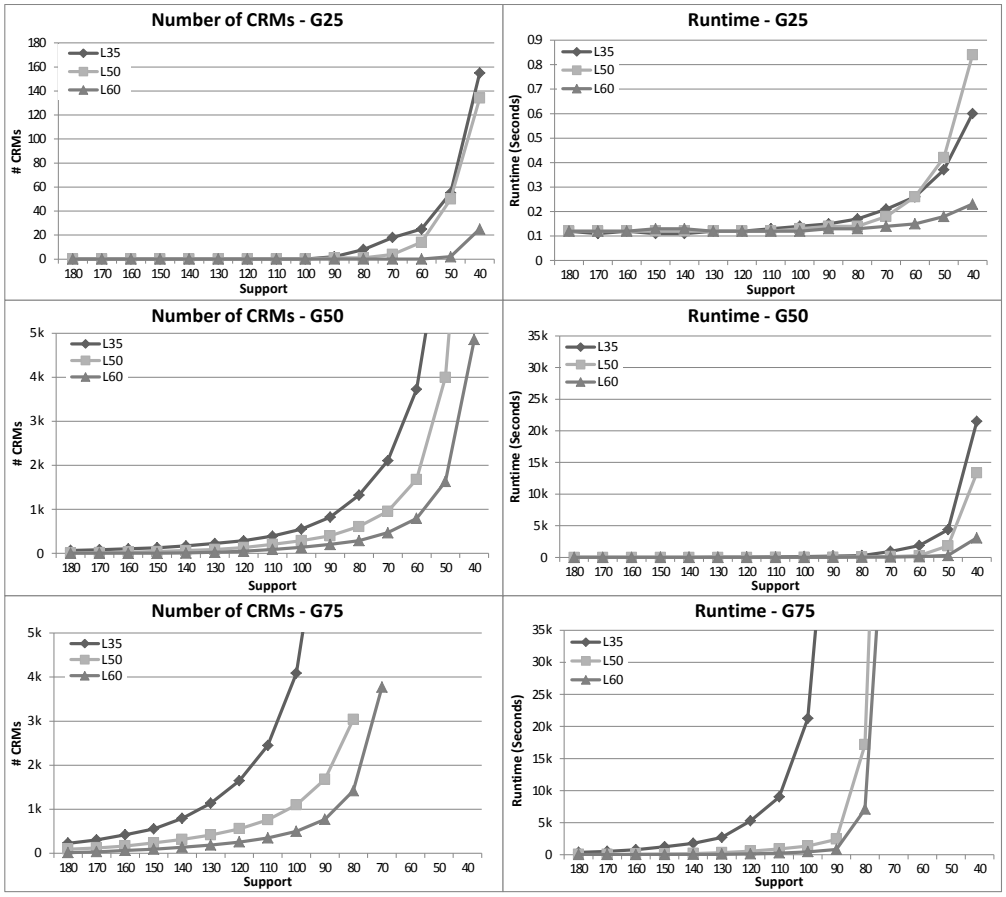


Fig. 13. Performance of CRMminer$_x$ for different versions of the $DBLP$ dataset. For all experiments, β is 0.60, m_{min} is 4, k_{min} is 3 and k_{max} is 10.

networks. First, we generated three datasets $L35$, $L50$, and $L60$ containing different number of labels by clustering the publication titles into 35, 50, and 60 groups respectively. Then for each of the three datasets, we generated three networks $G25$, $G50$, and $G75$. The $G25$ dataset was generated by removing all the relations of an author if he/she had co-authored with more than 25 other authors in a single year. Similarly, for $G50$ and $G75$, the maximum co-authorship threshold per year was set to 50 and 75 correspondingly. The total number of authors remained the same while the total number of edges increased from $G25$ to $G75$. Thus, the density is lowest in $G25$ and highest in $G75$. The number of edges in $G25$, $G50$, and $G75$ datasets are 2522412, 3973710 and 4554474 correspondingly.

Figures 12 and 13 present the results using these datasets from CRMminer and CRMminer$_x$, respectively. The results are displayed in increasing order of density from $G25$ to $G75$ and each graph shows the impact of support threshold in finding CRMs. Overall, as the dataset density increases, the number of discovered CRMs increases. This is expected, since the number of patterns in a denser network will most likely be greater. In addition, as the label diversity increases, the number of discovered CRMs decreases. This is because by increasing the number of labels the effective support of a CRM decreases, leading to fewer CRMs. Finally, these results show that a signif-

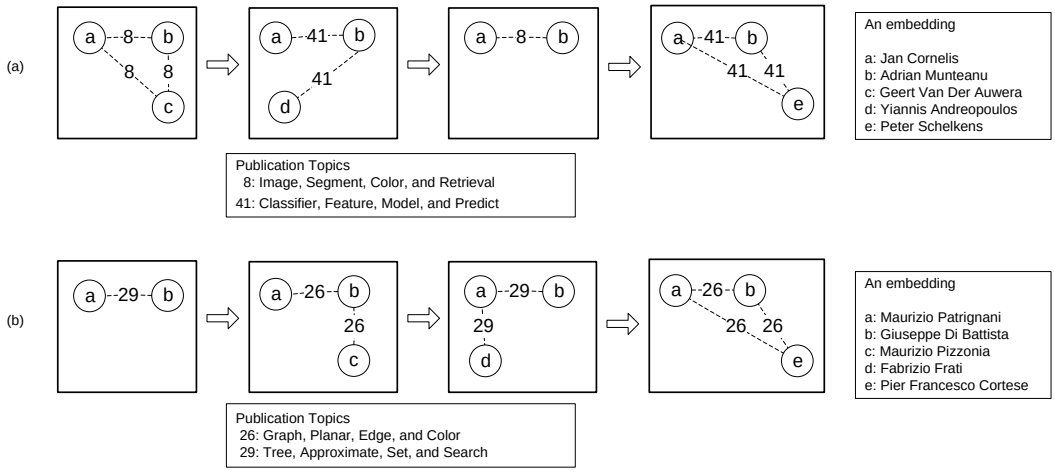


Fig. 14. Two CRMs capturing co-authorship patterns. The edge labels represent the domain or subject of the publications the authors were involved together. The vertices are labeled to show the changes in relations between the nodes. These CRMs are collected using $\phi = 90$ and $\beta = 0.60$.

icant improvement in runtimes between CRMminer and CRMminer$_x$ is found when comparing the same graph sets in Figures 12 and 13. For example, a comparison of results between the $G75$ datasets shows that CRMminer$_x$ is able to complete execution for lower thresholds in about $2 - 30$ times faster than CRMminer. In addition, the increase in runtimes as the support decreases is linear and proportional to the output space. Hence, CRMminer$_x$ is an excellent option for processing the dense graphs with low label diversity.

7.2. Qualitative Analysis

7.2.1. DBLP Case Studies. For the $DBLP$ dataset, the yearly co-authorship relations among the authors are divided into 50 clusters based on the title of the papers (Section 6.1). To rank the discovered CRMs, we use the cosine similarity between the centroids of the clusters, referred as *topic similarity*, that ranges from 0.02 to 0.52. For each CRM, we determine a score by calculating the average topic similarity based on all topic transitions (i.e., edge label changes) between two consecutive motifs of the CRM. The CRMs containing the least score ranks the highest. This ranking is designed to capture the frequent co-authorship relational changes that are thematically the most different. Note that the clusters are based on the publication titles and we use the most frequent words that belong to a cluster to describe the topic it represents.

Two of the high-ranked CRMs are shown in Figure 14. The first CRM shows the periodic changes in research topics represented as 8 and 41 and the topic similarity between these topics is 0.22. The CRM captures the periodic transitions of the relations as author a and b collaborate with other authors c, d, and e over the time. The second CRM shows the periodic changes in research topics represented as 29 and 26 and the topic similarity between these topics is 0.20. The CRM captures similar the periodic transitions of the relations as author a and b collaborate with other authors c, d, and e over the time.

7.2.2. Sales Case Studies. For the $Sales$ dataset, we ranked the discovered CRMs according to their size (i.e., the number of edges) and larger CRMs are ranked higher. We

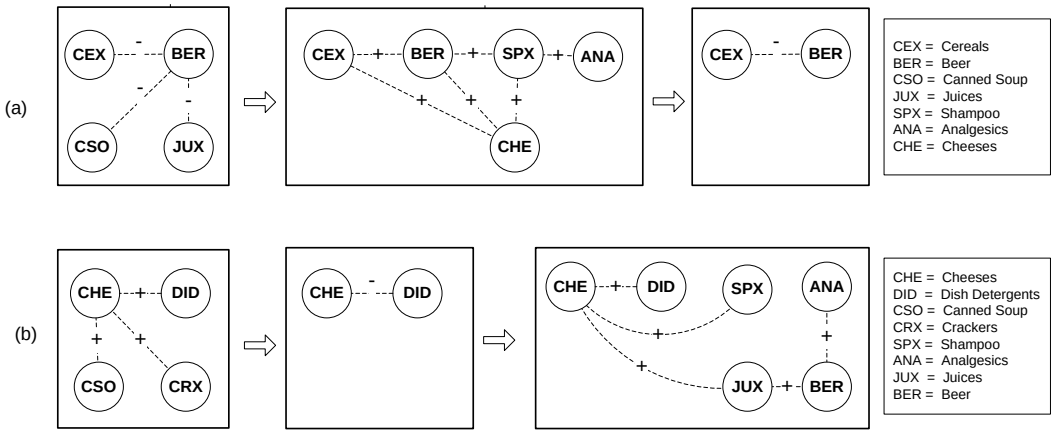


Fig. 15. Two CRMs capturing store sales patterns. The edge labels $+$ and $-$ correspond to positive and negative correlation between entities. These CRMs are collected using $\phi = 40$ and $\beta = 0.40$.

want to capture the sales pattern where a large group of products either gain or lose their sales correlation (i.e., tendency of being sold together). In Figure 15(a), a CRM of size 8 is presented from the Sales network capturing correlated product groups at a certain period and the changes in their relations. At first, the sale of CEX, CSO, and JUX product groups seem to be negatively correlated with BER. In the next motif, the sale of five product groups becomes positively correlated. Based on the relations among the nodes, the itemsets (CEX, BER, CHE) and (BER, SPX, CHE) are strongly correlated. In the last motif, the relation between the items changed as (CEX, BER) became negatively correlated. Based on the details of the different embeddings and motif occurrence period, it seems that during the holiday periods (i.e., thanksgiving, Christmas) all the product groups are positively correlated.

In Figure 15(b), another CRM of size 7 is presented. In this case, product groups are all positively correlated at the beginning. (DID, CSO, CRX) are positively correlated with CHE. In the next motif, the sale of (CHE, DID) becomes negatively. However, at the last period, (DID, JUX, SPX) product groups become positively correlated with CHE. The sale of (JUX, ANA) are also positively correlated with BER. Note that one can focus on understanding why the relation between CHE and DID changed and try to plan for some actions that stores can take to change the sales of those products.

7.2.3. Genentech Case Studies. For the GT dataset, the discovered CRMs display the dynamics of various process parameters during the cell culture process. As it is difficult to understand the relations between the nodes (i.e., parameters) without sufficient domain knowledge in Bio-Chemical processes, we decided to analyze the discovered CRMs by using them as features to discriminate the production runs. Out of the 247 production runs included in the GT dataset, based on the quality of the yields, 48 of the runs are labeled as *Good*, 48 of the runs are labeled as *Bad*, and the remaining were not labeled. To understand the class distribution (i.e., Good or Bad) of the discovered CRMs, we analyzed the embeddings of the discovered CRMs from three different experiments using ϕ of 90, 80 and 70. Since we have 96 labeled runs, we do not count the embeddings that belong to unlabeled runs. Figure 16 shows the class distribution of the embeddings for the discovered CRMs. It shows that the CRMs were present

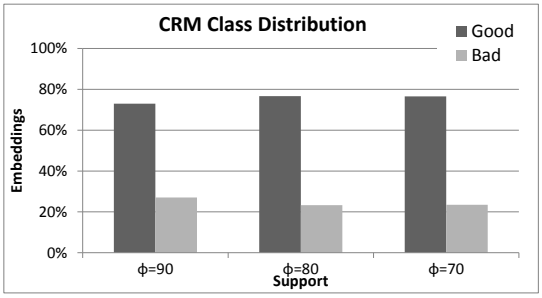


Fig. 16. A distribution of the CRM embeddings. The Good class represents the production runs with high yield and the Bad class represents with poor yield. CRMs were collected using ϕ = (90, 80 and 70), $\beta = 0.60$, $m_{min} = 3$, $k_{min} = 4$, and $k_{max} = 10$.

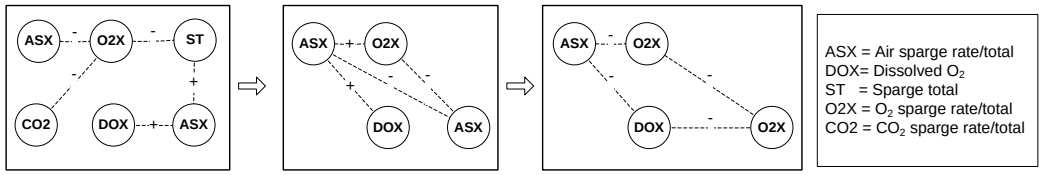


Fig. 17. A CRM capturing a cell culture bioprocess pattern. The edge labels $+$ and $-$ correspond to positive and negative correlation between entities. This CRM is collected using $\phi = 90$ and $\beta = 0.50$.

mostly as part of the high yield runs, since more than 70% of the embeddings belong to the Good class. The class representation is consistent for different support parameters. These results suggest that there is a consistency of what makes some thing good but runs can go bad for many reasons. Note that using such information, if we can determine the low yield runs at the early stage of experiment, we can terminate the experiment and save a lot of resources.

Figure 17 shows a CRM of size 10 from the GT network capturing the changes in entity correlation based on their recorded measurements at a particular stage of the cell culture process. Based on the embedding details, we noticed that the motif spans are mostly non overlapping. This indicates that the changes in the entity relations occur at different stages of the cell culture process based the experimental/process environment setup.

8. CONCLUSION & FUTURE DIRECTIONS

In this paper, we introduced coevolving relational motifs to represent patterns that change in a consistent way over time in a dynamic network and presented an algorithm to efficiently find all frequent coevolving relational motifs. The algorithm can be used to discover unknown coordination mechanisms in a system by identifying the patterns that evolve and move in a similar and highly conserved fashion in the dynamic networks. The experimental evaluation using multiple real world datasets show that CRMminer is able to discover CRMs from all datasets and CRMminer$_x$ scales better than CRMminer for large and dense dynamic networks. Further, the qualitative analysis shows that the discovered patterns capture important information and can be used as differentiating features for other mining problems.

There are a number of ways the CRM definition can be modified to address important special classes of CRMs. First, require that the set of identified CRMs is non-

redundant in the sense that no CRM is a subsequence of another CRM. This can substantially reduce the number of identified CRMs without loss of information. Second, the occurrences of the motifs are temporally synchronized, i.e., the snapshot in which each CRM s motif occurs is the same. Third, mine the CRMs that do not contain an anchor. Finally, require that the set of identified CRMs to be closed; i.e., at least one of its extensions occurs fewer times.

ACKNOWLEDGMENTS

This work was supported in part by NSF (IOS-0820730, IIS-0905220, OCI-1048018, CNS-1162405, and IIS-1247632) and the Digital Technology Center at the University of Minnesota. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

REFERENCES

Rezwan Ahmed and George Karypis. 2012. Algorithms for mining the evolution of conserved relational states in dynamic networks. *Knowledge and Information Systems* (2012), 1–28.

Tatsuya Asai, Kenji Abe, Shinji Kawasoe, Hiroki Arimura, Hiroshi Sakamoto, and Setsuo Arikawa. 2002. Efficient substructure discovery from large semi-structured data. In *Proc. of the 2nd SIAM Symposium on Data Mining*. 158–74.

T.Y. Berger-Wolf and J. Saia. 2006. A framework for analysis of dynamic social networks. In *ACM KDD*. 523–528.

M. Berlingerio, F. Bonchi, B. Bringmann, and A. Gionis. 2009. Mining graph evolution rules. *Machine Learning and Knowledge Discovery in Databases* (2009), 115–130.

Karsten M. Borgwardt, Hans-Peter Kriegel, and Peter Wackersreuther. 2006. Pattern Mining in Frequent Dynamic Subgraphs. In *IEEE ICDM*. 818–822.

Danah Boyd and Nicole Ellison. 2007. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Comm.* 13, 11 (October 2007).

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30, 1-7 (1998), 107–117.

Björn Bringmann and Siegfried Nijssen. 2008. What is frequent in a single graph? *Advances in Knowledge Discovery and Data Mining* (2008), 858–863.

L. Cerf, T. Nguyen, and J.F. Boulicaut. 2009. Discovering relevant cross-graph cliques in dynamic networks. *Foundations of Intelligent Systems* (2009), 513–522.

Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. 2006. Evolutionary clustering. In *ACM KDD*. 554–560.

P Chen and Sidney Redner. 2010. Community structure of the physical review citation network. *Journal of Informetrics* 4, 3 (2010), 278–290.

W. W. Cohen. 2005. Enron email dataset. Website. (2005). http://www.cs.cmu.edu/~enron/.

Elise Desmier, Marc Plantevit, Céline Robardet, and Jean-François Boulicaut. 2012. Cohesive Co-evolution Patterns in Dynamic Attributed Graphs. In *Discovery Science*. Springer, 110–124.

D. Duan, Y. Li, Y. Jin, and Z. Lu. 2009. Community mining on dynamic weighted directed graphs. In *Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management*. ACM, 11–18.

Thomas L. Friedman. 2005. *The world is flat: A brief history of the twenty-first century*. Farrar, Straus & Giroux.

Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics reports* 519, 3 (2012), 97–125.

H. Hu, X. Yan, H. Yu, J. Han, and X.J. Zhou. 2005. Mining coherent dense subgraphs across massive biological networks for functional discovery. In *ISMB*. Ann Arbor, MI, 213–221.

Xiaohua Hu. 2005. Mining and analysing scale-free protein protein interaction network. *Int. Journal of Bioinformatics Research and Applications* 1, 1 (2005), 81–101.

Jun Huan, Wei Wang, and Jan Prins. 2003. Efficient Mining of Frequent Subgraph in the Presence of Isomophism. In *IEEE ICDM*.

A. Inokuchi and T. Washio. 2008. A fast method to mine frequent subsequences from graph sequence data. In *IEEE ICDM*. 303–312.

A. Inokuchi and T. Washio. 2010. Mining frequent graph sequence patterns induced by vertices. In *Proc. of 10th SIAM Intl. Conf. on Data Mining*. 466–477.

Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. 2000. An Apriori-based algorithm for mining frequent substructures from graph data. In *Proc. of the 4th European Conf. on Principles of Data Mining and Knowledge Discovery*. Springer-Verlag, 13–23.

Ruoming Jin, Scott McCallen, and Eivind Almaas. 2007. Trend Motif: A Graph Mining Approach for Analysis of Dynamic Complex Networks. In *IEEE ICDM*. 541–546.

Yehuda Koren, Stephen C. North, and Chris Volinsky. 2007. Measuring and extracting proximity graphs in networks. *ACM Trans. Knowl. Discov. Data* 1, 3 (2007), 12.

M. Koyutürk, Y. Kim, S. Subramaniam, W. Szpankowski, and A. Grama. 2006. Detecting conserved interaction patterns in biological networks. *Journal of Computational Biology* 13, 7 (2006), 1299–1322.

S. Kramer, L. De Raedt, and C. Helma. 2001. Molecular Feature Mining in HIV Data. In *ACM KDD*.

Michihiro Kuramochi and George Karypis. 2004. An Efficient Algorithm for Discovering Frequent Subgraphs. *IEEE TKDE* 16, 9 (2004), 1038–1051.

M. Lahiri and T.Y. Berger-Wolf. 2008. Mining periodic behavior in dynamic social networks. In *IEEE ICDM*. 373–382.

Steve Lawrence, C. Lee Giles, and Kurt Bollacker. 1999. Digital Libraries and Autonomous Citation Indexing. *Computer* 32, 6 (1999), 67–71.

Huong Le, Santosh Kabbur, Luciano Pollastrini, Ziran Sun, Keri Mills, Kevin Johnson, George Karypis, and Wei-Shou Hu. 2012. multivariate analysis of cell culture bioprocess data–Lactate consumption as process indicator. *Journal of Biotechnology* (2012).

Michael Ley. 2008. DBLP, Computer Science Bibliography. Website. (2008). http://www.informatik.uni-trier.de/\~ley/.

Bing Liu. 2007. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer.

Xiaoming Liu, Johan Bollen, Michael L. Nelson, and Herbert Van de Sompel. 2005. Co-Authorship Networks in the Digital Library Research Community. *Information Processing and Management* 41, 6 (2005), 1462–1480.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Ping, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2001a. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *Proceedings 2001 International Conference on Data Engineering*.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2001b. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. In *ICDE*. 215–224. citeseer.nj.nec.com/pei01prefixspan.html

Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On mining cross-graph quasi-cliques. In *ACM KDD*. 228–238.

Matjaž Perc. 2010. Growth and structure of Slovenias scientific collaboration network. *Journal of Informetrics* 4, 4 (2010), 475–482.

Matjaž Perc and Attila Szolnoki. 2010. Coevolutionary gamesa mini review. *BioSystems* 99, 2 (2010), 109–125.

C. Robardet. 2009. Constraint-based pattern mining in dynamic graphs. In *IEEE ICDM*. 950–955.

B. Schwikowski, P. Uetz, and S. Fields. 2000. A network of protein-protein interactions in yeast. *Nature Biotechnology* 18, 12 (December 2000), 1257–1261.

Lei Tang, Huan Liu, Jianping Zhang, and Zohreh Nazeri. 2008. Community evolution in dynamic multimode networks. In *ACM KDD*. 677–685.

B. Wackersreuther, P. Wackersreuther, A. Oswald, C. B
"ohm, and K.M. Borgwardt. 2010. Frequent subgraph discovery in dynamic networks. In *Proc. of the 8th Workshop on Mining and Learning with Graphs*. ACM, 155–162.

Xifeng Yan and Jiawei Han. 2002. gSpan: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 721–724.

Xifeng Yan, X. Jasmine Zhou, and Jiawei Han. 2005. Mining closed relational graphs with connectivity constraints. In *ACM KDD*. 324–333.

C.H. You, L.B. Holder, and D.J. Cook. 2009. Learning patterns in the dynamics of biological networks. (2009).

Mohammed J. Zaki. 2002. Efficiently mining frequent trees in a forest. In *ACM KDD*. 71–80.

Feida Zhu, Xifeng Yan, Jiawei Han, and S Yu Philip. 2007. gPrune: a constraint pushing framework for graph pattern mining. In *Advances in Knowledge Discovery and Data Mining*. Springer, 388–400.

Received Month Year; revised Month Year; accepted Month Year