Online Spectral Clustering on Network Streams

By

Yi Jia

Submitted to the graduate degree program in Electrical Engineering and Computer Science
and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Committee members

Jun Huan, Chairperson

Swapan Chakrabarti

Jerzy Grzymala-Busse

Bo Luo

Alfred Tat-Kei Ho

Date defended: _____

The Dissertation Committee for Yi Jia certifies
that this is the approved version of the following dissertation :

Online Spectral Clustering on Network Streams

Jun Huan, Chairperson

Date approved: _____

Abstract

Graph is an extremely useful representation of a wide variety of practical systems in data analysis. Recently, with the fast accumulation of stream data from various type of networks, significant research interests have arisen on spectral clustering for network streams (or evolving networks). Compared with the general spectral clustering problem, the data analysis of this new type of problems may have additional requirements, such as short processing time, scalability in distributed computing environments, and temporal variation tracking.

However, to design a spectral clustering method to satisfy these requirement certainly presents non-trivial efforts. There are three major challenges for the new algorithm design. The first challenge is online clustering computation. Most of the existing spectral methods on evolving networks are off-line methods, using standard eigensystem solvers such as the Lanczos method. It needs to recompute solutions from scratch at each time point. The second challenge is the parallelization of algorithms. To parallelize such algorithms is non-trivial since standard eigen solvers are iterative algorithms and the number of iterations can not be predetermined. The third challenge is the very limited existing work. In addition, there exists multiple limitations in the existing method, such as computational inefficiency on large similarity changes, the lack of sound theoretical basis, and the lack of effective way to handle accumulated approximate errors and large data variations over time.

In this thesis, we proposed a new online spectral graph clustering approach with a family of three novel spectrum approximation algorithms. Our algorithms

incrementally update the eigenpairs in an online manner to improve the computational performance. Our approaches outperformed the existing method in computational efficiency and scalability while retaining competitive or even better clustering accuracy. We derived our spectrum approximation techniques GEPT and EEPT through formal theoretical analysis. The well established matrix perturbation theory forms a solid theoretic foundation for our online clustering method. We facilitated our clustering method with a new metric to track accumulated approximation errors and measure the short-term temporal variation. The metric not only provides a balance between computational efficiency and clustering accuracy, but also offers a useful tool to adapt the online algorithm to the condition of unexpected drastic noise. In addition, we discussed our preliminary work on approximate graph mining with evolutionary process, non-stationary Bayesian Network structure learning from non-stationary time series data, and Bayesian Network structure learning with text priors imposed by non-parametric hierarchical topic modeling.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graph is an extremely useful representation of a wide variety of practical systems [270, 124, 201, 9, 42, 7, 64, 238, 18, 49, 193, 155, 129, 255], which has the advantage to uncover explicit or implicit dependency relations between structural components (spatial or temporal) among data [58]. In many real-world graphs, communities usually represent key behavioral or functional units within graphs [151]. Such finding has encouraged researchers to contribute considerable efforts on community detection in graphs [56, 33, 91].

In data analysis, graph communities are typically described as groups of nodes with denser connections inside groups and sparser connections between groups and the partitioning of a graph into communities is called graph clustering [231] or community detection in graphs [81]. Their plentiful applications could be found in image processing [237, 200], retailers' co-purchasing analysis [56, 93], academic collaboration and citation network analysis [80, 71, 198], real-life local social network analysis [71, 73, 82], food web analysis of animals [93, 82], animals' social network analysis [82, 198], web searching [80, 46],speech processing [18], and document analysis [198].

The existing graph clustering methods could be summarized into two classes [231, 44]: divisive clustering and agglomerative clustering. The divisive clustering is top-down methods to iteratively divide the larger vertex sets into smaller sets [152, 150, 29, 249, 237].

The agglomerative clustering is bottom-up methods that start with single vertex sets and recursively merge smaller sets into larger sets[56, 73]. The divisive clustering methods are further divided into five categories: cut based clustering [78, 57, 150, 80, 46], multilevel clustering [152], the betweenness based clustering [83, 29, 92, 93], random walk based clustering [249, 241], and spectral clustering [39, 15, 237, 104, 200, 198, 18]. The typical agglomerative algorithms are the modularity measure [199] based approaches [71, 56, 73]. It was showed in the work of Frivanek et al. [159] that if the levels of dendrogram, the hierarchical cluster tree, is more than 3, all these clustering problems are NP-hard .

Another important dichotomy of graph clustering techniques is heuristic and spectral [196]. The previous discussed categories excluding spectral clustering methods all drop into the heuristic domain. Different from most purely experiences based heuristic clustering approaches to trade optimality for speed [213], spectral clustering approaches have the advantage of providing lower/upper bounds for the minimization/maximization of various graph partitioning objective functions [196], such as normalized cuts [237] and ratio-cut measure [45].

Recently, with the fast accumulation of stream data from various type of networks, significant research interests have arisen on spectral clustering for evolving networks (or network streams). Researchers have used spectral clustering techniques to investigate evolving graphs in various applications, such as, online bloggers' friendship networks [203, 48, 49], simulated bird flocking movement data [267], MIT campus student friendship networks [267], and academic publication networks [247, 99].

We summarized the existing work in spectral clustering problems on evolving graph into two parts: incremental spectral clustering [248, 158, 203] and evolutionary spectral clustering [49, 266, 247, 99].

Incremental clustering are usually used to handle two types of clustering tasks [266]: (i) that sequentially clusters incoming new data points that are each observed once, known as data stream clustering [40]; (ii) that continuously clusters data points that are each

observed at multiple time points. The incremental clustering tasks are mainly focusing on high computational efficiency.

Evolutionary spectral clustering aims to discover clusters in a sequence of clustering tasks from evolving graph data [37]. It is designed specifically for evolving graph data with slowly drifting clustering boundaries and use temporal smoothness functions to eliminate short-term noises and improve the clustering accuracy.

In this dissertation, we are focusing on task (I) in incremental spectral clustering problems. Compared with the general spectral clustering problem, the data analysis of this new type of problems may have the following additional requirements.

- It may need to process large scale graph stream in real-time manner with short enough time overheads.
- It may need to alleviate the computational concern in distributed computing environments.
- It may need to track large temporal variations over time.

The applications containing those three aspects are plentiful. We discuss three examples below.

- Online Social Networks (OSNs), such as Facebook, Twitter, and Linkedin, have made significant impacts on reshaping the social connections and mind-share among people. Spectral clustering techniques provide a very useful tool to track the shifting of virtual communities and their members' activity [263, 162]. However, the gigantic number (millions) of users and their interactions pose great computational challenges to the classical clustering methods. and their mutual similarity information evolves continuously over time.
- Understanding Internet topologies is critically important for ISPs (Internet Service Providers) to enable efficient network management and security monitoring. Currently

such analysis relies on data information extraction on Internet Autonomous Systems (ASs) and exploring the properties of associated graphs on the AS-levels [245]. Spectral analysis has been widely used to understand the network behavior by grouping ASs [94, 194, 245]. However, the increasing large number of hosts in the backbone networks poses significant challenges to the fine-granularity analysis of continuously evolving network topologies [268]. For ISPs, with always limited computational resources, continuously expanding network scales, and highly preferred fast real-time processing feature, it is important to improve the computational performance of clustering methods to analyze ASs.

- A typical recommender system aggregates and directs recommendations to appropriate recipients based on the opinions and behaviors of other people with the similar interests [225, 30]. Recently recommender systems become extremely popular in E-commerce, such as Amazon.com and Ebay.com, and online Social Networks, such as Facebook.com. Spectral clustering demonstrated itself a very useful tool to improve the output of recommender systems [3, 4]. However, the choice of products or contents of interests available to people are gigantic, and their mutual similarity information evolves continuously over time. For example, the Amazon's person-to-person recommendation networks has half a million product nodes [173]. With always limited computational resources and continuously expanding network scales, to improve the computational performance of classical clustering methods is important.

Those applications with new properties bring us new challenges for data analysis. The first challenge is online clustering computation. Most of the existing spectral methods on evolving networks are off-line methods, using standard eigensystem solvers such as the Lanczos method [96], and need to recompute solutions from scratch at each time point.

The second challenge is the parallelization of algorithms. Scaling spectral clustering algorithms to very large graph data is challenging. To parallelize such algorithms is non-trivial since standard eigen solvers are iterative algorithms and the number of iterations can

not be predetermined.

The third challenge is the limited existing work. Currently the existing work of online spectral clustering on large graph streams is very limited. Although the incremental spectral clustering method proposed by Ning et al. [203] has been designed to improve the computational performance of spectral clustering for large network streams, there exists multiple limitations in Ning's work, such as computational inefficiency on large similarity changes, the lack of sound theoretical basis, and the lack of effective way to handle accumulated approximate errors and large data variations over time.

To address these challenges, we proposed an online spectral clustering method ISSUER (**I**ncremental **S**pectral clu**S**tering based on matrix pert**U**rbation th**E**o**R**y) with three novel spectrum approximation algorithms: FOA (First Order Approximation), GEPT (General Eigen Perturbation Theory) and EEPT (Enhanced Eigen Perturbation Theory). The FOA algorithm is based on the first order spectrum approximation. Its eigenvalue approximation is consistent with the results of the eigenvalue perturbation method based on Gerschgorin's Theorem [89] used in GEPT. The GEPT algorithm follows Wilkinson's work [264] to create sharp bounds for perturbed eigenvalues by shrinking the Gerschgorin disks [89]. It used Stewart's invariant subspace perturbation theory [244] to approximate eigenvectors. By observing the concerns of GEPT on computational costs and accumulated errors, we proposed our third approach EEPT by re-investigating the theoretical formalization of Stewart's work [244] under the context of evolutionary scenario. EEPT solves both eigenvalue and eigenvector approximation problem on the same theoretic basis of the invariant subspace perturbation.

1.1 Contributions

The major contributions of this thesis are as follows.

- We proposed a family of three novel spectrum approximation algorithms. Our algorithms incrementally update the eigenpairs in an online manner to improve the

computational performance. Our approaches outperform the existing method [203] in computational efficiency and scalability while retaining competitive or even better clustering accuracy.

- We derived our spectrum approximation techniques GEPT and EEPT through formal theoretical analysis. The well established matrix perturbation theory forms a solid theoretic foundation for our online clustering method.

- We facilitated our clustering method with a new metric to track accumulated approximation errors and measure the short-term temporal variation. The metric not only provides a balance between computational efficiency and clustering accuracy, but also offers a useful tool to adapt the online algorithm to the condition of unexpected drastic noise.

- In our preliminary work, we developed multiple novel algorithms on evolving data. We proposed a novel generative frequent subgraph mining method that used a stochastic matrix to score label distortions in matching a subgraph pattern to a graph. We devised our two novel non-stationary Bayesian Network structural learning algorithms from non-stationary time series data. In addition, in the context of limited available samples but with additional unstructured text data, we proposed a new BN structure learning method with text priors imposed by non-parametric hierarchical topic trees to improve the prediction accuracy.

1.2 Thesis Organization

This thesis is organized in the following way. In Chapter 2, we will outline the backgrounds of online spectral clustering problem and our previous research work. In Chapter 3, we will present a novel graph database mining method called APGM (APproximate Graph Mining) to mine useful approximate subgraph patterns from noisy protein graph databases.

In Chapter 4, we will show a new non-stationary Dynamic Bayesian Network structure learning method based on RJMCMC, domain knowledge on potential regulator detection, and a flexible lag choosing mechanism to predict biological gene regulatory networks from microarray data. In Chapter 5, we will introduce another novel non-stationary Dynamic Bayesian Network structure learning method based on Markov Chain Monte Carlo (MCMC) and Perfect Simulation techniques to model the time varying gene regulatory networks in the same data sets studied in Chapter 4. In Chapter 6, we will present a new Bayesian Network structure learning approach with text priors that is investigated and demonstrated in psychometric domain. In Chapter 7, we will show how to design various eigen system approximation techniques to propose our new online spectral clustering method, ISSUER. Finally, in Chapter 8, we conclude our thesis and discuss our future work.

Chapter 2

Background

In this chapter, we will first introduce the backgrounds of our online spectral clustering problems and of our previous research work.

2.1 Unsupervised Learning of Graphs

Machine learning may be divided into two categories: supervised learning and unsupervised learning. While the goal of supervised learning is to find a mapping function from data points to labels (categories or real numbers) to minimize a chosen loss function, unsupervised learning aims to find patterns (or structures) over or beyond the pure noises among data to build a new, compact and more informative representation of the input [90]. The mined patterns could be used to help a supervised problem and get validated or could be applied to organize and visualize the data [25].

The models dropping into the category of unsupervised learning are very plentiful, for example, clustering [136, 23, 218], Principle Component Analysis (PCA) [138, 137, 236], Mixture Model [17, 220], Graphical Statistical Model [146, 236], Hidden Markov Model (HMM) [216, 21], and many subfields of Data Mining, such as, substructure mining [270, 124, 112], association rule mining [35, 95], frequent sequence mining [272, 258], ranking [206, 156], etc.

The representation of graphs is used to encode the networks in many applications of machine learning, e.g. biology [155, 129, 278, 171], chemistry [270, 201], drug design [98], social science [18, 49], document management [7, 64], security [42], image processing [238, 18], and education [239, 31].

Unsupervised learning of graphs aims to mine patterns from graphs, e.g. frequent subgraph mining [270, 124, 201], aggregated synopsis graph mining on graph stream [9, 42], cluster centroid learning in the clustering problem on graphs [7, 64], and graph clustering on graphs [152, 150, 29, 249, 238], etc..

In this thesis, we are focusing on two types of models in unsupervised learning of graphs: frequent subgraph mining and graph spectral clustering. Frequent subgraph mining is an active research topic to detect subgraph patterns that appear in a graph database with frequency no less than a user-specified threshold [112]. Spectral clustering is a clustering task to transfer the minimization/maximization of a graph partitioning or cut problem into the spectral analysis problem of different variants of graph Laplacians [196].

2.2 Graph Spectral Clustering

2.2.1 Representation of Graphs

In this section, we will give the formal definitions of graphs and other correlated important terms.

A **weighted undirected graph** G is a 3-tuple $G = (V, E, W)$ where V is the set of vertices of G and $E \subseteq V \times V$ is the set of undirected edges of G with $(u, v) \equiv (v, u) : u, v \in V$. $W : V \times V \to R^+ \cup \{0\}$ is the function assigning a non-negative real value $W(l)$ to each node pair $l \in V \times V$. If a link $l \in E$, then $W(l) > 0$, otherwise $W(l) = 0$.

The **adjacency matrix** $A(G)$ of a weighted undirected graph G with n nodes is an $n \times n$ matrix, where each entry $a_{i,j} = W(i, j)$. A is a symmetric matrix. The **graph Laplacian matrix** L for a given G is defined as $L(G) = D(G) - A(G)$, where $D(G)$ is a diagonal matrix

with $d_{i,i} = \sum_j a_{i,j}$. The **transition matrix** $B(G) = D^{-1} \times A(G)$.

2.2.2 Related Work on Spectral Clustering

Spectral graph clustering methods can be classified into two groups [196]: recursively two-way approaches [237, 198] and direct$k-$way approaches [39, 15, 104, 200, 198].

Shi et al. in 1999 [237] proposed a two-way graph clustering method based on normalized cuts. Given a cut $(S, V\backslash S)$ of a graph $G = (V, E, W)$, a normalized cut is defined as $Ncut(S, V\backslash S) = \frac{C(S,V\backslash S)}{assoc(S,V)} + \frac{C(V\backslash S,S)}{assoc(V\backslash S,V)}$, where $assoc(S, V) = \sum_{u\in S,t\in V} W(u, t)$ and $assoc(V\backslash S, V) = \sum_{u\in V\backslash S,t\in V} W(u, t)$. By transferring minimum normal cut problem into a Rayleigh quotient and further relaxing one of its constraints into real value domain, Shi's method provided a solution of second smallest eigenvector in a generalized eigenvalue system.

Newman in 2006 [198] provided a two-way clustering method based on modularity measure [199]. The modularity function is defined as $Q = \frac{1}{2m}\sum_{i,j}(W(i, j) - P(i, j))\delta(g_i, g_j)$. m is the number of edges in the graph. $P(i, j)$ is the expected weight of the edges between nodes i and j in the null graph model. g_i and g_j are the groups that nodes i and j belongs to. $\delta(g_i, g_j)$ is equal to 1 when $g_i = g_j$ and otherwise -1. Newman's method used the configuration models proposed in [55] as the null model to simulate the right-skewed degree distributions found in real-world networks [20]. His method provided a solution to the maximum modularity problem as the eigenvector with the largest eigenvalue in the modularity matrix B, where $B(i, j) = W(i, j) - P(i, j)$.

Chan et al. in 1994 [39] proposed a spectral $k-$way graph partitioning method based on ratio-cut measure [45]. In their method, they established a connection between $k-$way ratio-cut partitioning and Hall's work on generalized weighted quadratic placement [111]. They approximated the ratioed assignment matrix R with the eigenvectors V with $k-$smallest eigenvalues of the graph Laplacian and recovered the objective partition matrix P from VV^T. Considering the expensive computation on P, they used directional cosine of two vectors to evaluate the similarity of vertices. They provided a heuristic to form the partition

without calculating the whole P matrix.

Alpert et al. in 1999 [15] proposed a spectral $k-$way partitioning heuristic based on minimum cut measure. their method used a graph's eigenvectors to construct a geometric representation of the graph and reduced the original graph partitioning problem into a vector partitioning problem. The eigenvectors of the graph Laplacian were used as the coordinates of vectors representing vertices.

Gu et al. in 2001 [104] proposed a spectral $k-$way graph clustering method based on Min-max cut measure. Given a $k-$way cut $\Pi = (S_1, S_2, \cdots, S_k)$, Min-max cut is defined as $Mcut(\pi) = \sum_{i=1}^{k} \frac{C(S_i, V \setminus S_i)}{C(S_i, S_i)}$, where $C(S_i, S_i) = \sum_{u \in S_i, v \in S_i} W(u, v)$. The min-max aims at minimizing inter-cluster similarities while maximizing intra-cluster similarities. Their method provided a solution to their Min-max cut problem as any orthonormal basis of the subspace spanned by the eigenvector pertaining to the $k-$largest eigenvalues of the normalized Laplasian.

Meila et al. in 2001 [187] proposed a spectral $k-$way graph clustering approach based on random walks. They established the relation between the eigenpair of the stochastic matrix and the generalized eigenpair of graph Laplacian. Their method solved the k-way graph clustering problem by selecting the eigenvectors $[x^1, \cdots, x^k]$ corresponding to k-largest eigenvalues and applying k-means in the $k-1$ dimensional space defined by $[x^2, \cdots, x^k]$.

Ng et al. in 2002 [200] provided a spectral $k-$way clustering algorithm based on the nCut measure. Given a $k-$way cut $\Pi = (S_1, S_2, \cdots, S_k)$, The nCut measure is defined as $nCut(\pi) = \frac{1}{2} \sum_{i=1}^{k} \frac{C(S_i, V \setminus S_i)}{vol(S_i)}$, where $vol(S_i) = \sum_{v_j \in S_i} \sum_{l=1}^{n} W(j, l)$ and $n = |V|$. Their method solved the nCut problem by applying K-means or any other general clustering algorithm on the eigenvector matrix corresponding to $k-$largest eigenvalues of the normalized Laplacian. In addition, they analyzed the stability of generated clusters by using matrix perturbation theory [244] by pointing out that the stability of the eigenvectors of a matrix is determined by the eigengap.

Newman in 2006 [198] extended his two-way method into $k-$way division. Newman's

method generalized its modularity measure into a $k-$way by incorporating the assignment matrix [39]. His method provided a solution of the maximum modularity problem as a $n \times (k+1)$ eigenvector matrix R corresponding to the positive eigenvalues of the modularity matrix. They found that the upper boundary of the number of clusters in the graph is the number $k+1$ of positive eigenvalues. With each row of R representing the features of each node, Newman transferred the graph partitioning problem into a vector partitioning problem on R.

2.3 Graph Clustering of Graph Streams

2.3.1 Graph Streams

Massive streams of graph data widely exist in a number of communication applications such as social networks [48, 247, 99, 5], telecommunication networks [267] and internet [203, 48, 49, 11]. Recently it has led to great research interests on various machine learning problems, such as outlier detection on graph streams [6, 10] and graph clustering on graph streams [203, 48, 247, 99]. Based on the definition of graphs in Section 2.2.1, we defined the term, graph streams, as follows.

An **evolving weighted undirected graph** $< G >$ is a sequence of weighted undirected graphs $< \ G_1, G_2, \ldots, G_T \ >$, where $V_{G_1} = \cdots = V_{G_T} = V$. For simplicity, in the remaining of the paper we write evolving graph to denote evolving weighted undirected graph. Correspondingly we define an **evolving matrix** $< A >$ as $< \ A_1, A_2, \ldots, A_T \ >$.

2.3.2 Related Work on Spectral Clustering of Graph Streams

The clustering problems in graph streams could be divided into two types: the traditional node clustering of individual graphs in a stream that we called graph clustering [203, 48] and the clustering of many different individual graphs in a stream [8, 10]. In this thesis, we are focusing on graph clustering in graph streams.

We summarized the related work on graph spectral clustering of graph streams into two parts: incremental spectral clustering and the related area, evolutionary spectral clustering.

2.3.2.1 Incremental Spectral Clustering

The incremental clustering tasks are mainly focusing on high computational efficiency. Although there is a large body of work on data stream incremental clustering other than spectrum analysis, such as, incremental hierarchical clustering [41], incremental micro-clustering [176], and incremental correlation clustering [184] and reference therein, the existing work on incremental spectral clustering is very limited only containing [203, 248, 158].

Ning et al. in 2007 [203] proposed an incremental spectral clustering approach based on similarity change operations on incidence matrix. Their method reduced the computational cost by incrementally updating the eigenvalues/vectors with single similarity change on incidence matrix. Their eigenvalue approximation is the first order Taylor approximation. Their perturbed eigenvectors is estimated based on their empirical finding that only the neighborhoods of the nodes connecting the changed edges contribute to the changes of perturbed eigenvectors. The computational gain of Ning's method is only obtained in the condition that a matrix perturbation related to affinity matrices or adjacency matrices of graphs consists of very limited number of similarity changes.

Valgren et al. in 2007 [248] proposed an incremental spectral clustering approach to address the applications where the entries in the affinity matrix are costly to compute. Their method repeatedly updated the cluster representative points of clusters. By evaluating the similarities between new points and existing cluster representatives, the new points in the stream are assigned into the existing or new clusters.

Kong et al. in 2011 [158] proposed an incremental spectral clustering that combines Ning's and Valgren's approaches. For each incoming new data point, they followed Valgren's work to recompute the cluster representatives. Based on the cluster representative points, their method created new matrices to build incidence matrix and applied Ning's similarity

operation to update eigen systems.

2.3.2.2 Evolutionary Spectral Clustering

Evolutionary clustering aims to use temporal smoothness functions to improve the clustering accuracy over time [37]. In recent years, this approach has greatly expanded in classical graph clustering algorithms, such as evolutionary agglomerative hierarchical clustering [108], evolutionary density-based clustering [154], and evolutionary spectral clustering [49, 247, 99, 266].

Chi et al. in 2007 [48] proposed a novel evolutionary spectral clustering approach that provided two temporal smoothing frameworks. The first framework, Preserving Cluster Quality (PCQ), considered how well the current model fits to the historical data. The second framework, Preserving Cluster Membership (PCM), considered how well the current model is consistent with the historical model. Based on these new criteria, they incorporated these temporal cost functions into the objective functions and provided relaxed solutions based on the spectrum analysis.

Tang et al. in 2008 [247] extended the evolutionary spectral clustering method to the multi-mode networks by introducing the temporal smoothness regularization in the block model. Their method encoded the interactions between two modes and clustering result of neighboring time stamps as general attributes. Hence, their method transferred the dynamic multi-mode graph clustering problem to the classical attribute-based clustering problem solved with singular value decomposition (SVD) and k-means.

A similar temporal cost function formalization as [48, 49] was used by Green et al. in 2011 [99] to propose an new Dynamic Spectral Co-Clustering algorithm (DSCC) that simultaneously groups clustered objects and their features over time. Their method incorporated the regularization term that calculated the difference between clustering centroid prediction and historical results. They followed Dhillon's work in [69] to solve it with truncated singular value decomposition.

Xu et al. in 2011 [266] proposed an new evolutionary clustering framework that adaptively estimated the optimal smoothing parameter using a shrinkage approach. They estimated the true proximity matrix by minimizing the squared Frobenius norm of the difference between the true proximity matrix and the smoothed proximity matrix. They adaptively estimated the forgetting factors of the regularization terms in an iterative way to calculate the sample means and variance in various cluster memberships.

2.4 Frequent Subgraph Mining

Frequent subgraph patterns are substructures that occur in a graph database (or a set of graphs) with frequencies equal to or above thresholds predefined by users. Frequent subgraph mining methods are the algorithms proposed to extract these frequent subgraph patterns from graph data.

Generally, in the frequent subgraph mining problem, we define a labeled graph as a 5-tuple $G = \{V, E, \Sigma_V, \Sigma_E, \lambda\}$ where V is the set of vertices of G and $E \subseteq V \times V$ is the set of undirected edges of G. Σ_V and Σ_E are (disjoint) sets of labels and labeling function $\lambda : V \to \Sigma_V \cup E \to \Sigma_E$ maps vertices and edges in G to their labels. A **graph database** D is a set of graphs.

The frequent subgraph mining problem suffers from intensive computation due to two reasons: (1) subgraph matching is known as an NP-complete problem and hence it is unlikely we will have a polynomial running time solution in general context with the exception of planar graphs, and (2) the total number of frequent patterns may grow exponentially in the number of graphs in a database and in the average size of the graphs in the database. Hence, many heuristics have been developed to speed up the subgraph mining procedures.

There are multiple ways to classify these frequent subgraph mining methods [165], such as the classification based on algorithmic approach (apriori and pattern growth), the classification based on search strategy (depth-first and breadth-first), the classification based on

input graphs (a single graph and a set of graphs), and the classification based on output patterns (complete pattern set and partial pattern set).

In this thesis, we will introduce the existing graph mining methods based on their search strategies.

2.4.1 Breadth-First-Search Frequent Subgraph Mining Methods

Breadth-First-Search (BFS) is a search strategy that explores a spanning tree of a graph with the order of being from left to right across levels.

Holder et al. in 1994 [119] provided a method called SUBDUE that used the Minimum Description Length (MDL) principle [226] to evaluate the quality of proposed substructure candidates. In SUBDUE, the MDL theory stated that the description length of the best substructures to represent the whole data should be the minimum. SUBDUE is a pattern growth based approach that extends the existing pattern candidates by attaching a single edge. For every size of patterns k, only a limited number of structure patterns with the best MDL scores are retained. SUBDUE does not claim the completeness of output pattern set.

Inokuchi et al. in 2000 [135] proposed a frequent induced subgraph mining algorithm called AGM (Apriori based Graph Mining). A subgraph $H = (V_H, E_H)$ of $G = (V_G, E_G)$ is induced if there existing a bijective function f such that for each $e \in E_G$ $f(e) \in E_H$. AGM encoded an identical induced subgraph with the canonical form of its adjacency matrix. The canonical forms of subgraphs were used to speed up the subgraph isomorphism. It used the join operation on two k sized similar patterns to propose a new $(k+1)$ size pattern that is called the Apriori based candidate enumeration.

Kuramochi et al. in 2001 [160] proposed a frequent subgraph mining algorithm FSG (Frequent SubGraphs). It used the adjacency-list representation to encode a graph. FSG enumerated new pattern candidates by adding an edge to the existing patterns. The canonical form of the adjacency-list were proposed to speed up the computation.

Kuramochi et al. in 2004 [161] proposed a fast subgraph mining algorithm HSIGRAM to

mine a complete pattern set on a single large graph. HSIGRAM used maximal independent set to find the edge-disjoint embeddings of the subgraphs. It iteratively enumerate the new candidates with the size $(k+1)$ based on the join operation on two k-size subgraphs that have a common $(k-1)$-size subgraph. HSIGRAM took the time-memory trade-off to only store partial information of the embeddings.

2.4.2 Depth-First-Search Frequent Subgraph Mining Methods

Depth-first search (DFS) is a search strategy that explores a spanning tree of a graph with the order of being down paths and being from left to right.

To target the computational issue of the join candidate enumeration operation and pruning false positives in AGM [135] and FSG [160], Yan et al. in 2001 [270] developed the first DFS graph mining algorithm gSpan (Graph based Substructure pAttern miNing). gSpan used DFS lexicographic order and minimum DFS code as a novel canonical labeling system to support the DFS search. It expanded the existing patterns by attaching a single edge to enumerate new candidates. It used Ullmans subgraph isomorphism matching algorithm and chose the pre-order traversal of the DFS code tree.

Borgelt et al. in 2002 [27] proposed a new frequent substructure mining algorithm to mine common patterns from molecule structures. It used candidate growth strategy to extend candidate size and avoided frequent re-embeddings.

Yan eta al. in 2003 [271] proposed a new frequent closed subgraph mining algorithm CloseGraph (Closed Graph pattern mining). A frequent subgraph pattern is closed if there are no super-patterns that has the same support values. The closed subgraph patterns are used to dramatically reduce the number of patterns in real-world application. CloseGraph was a DFS algorithm designed based on gSpan [270]. It used techniques such as equivalent occurrence and early termination to further prune the candidate search space. It outperformed gSpan in computational time with 4-10 factors in large graph databases (graphs with more than 32 edges).

Huan et al. in 2003 proposed a new graph mining algorithm called FFSM (Fast Frequent Subgraph Mining) [124]. FFSM used Canonical Adjacency Matrix (CAM) to encode the canonical form of a substructure pattern. FFSM considered the concerns of the multiple candidate generation issue in the join operation and the node attaching restriction issue in the extension operation. It provided two new candidate enumeration methods, FFSM Join and FFSM-Extension, on the suboptimal CAM tree. Through these strategies, it avoided the subgraph isomorphism checking.

Huan et al. in 2004 proposed a new frequent subgraph mining algorithm SPIN (SPanning tree based maximal graph mINing) [128]. In order to dramatically reduce the total number of substructure patterns, SPIN mined only maximal frequent subgraphs. SPIN combined the frequent spanning tree mining and frequent subgraph mining techniques. It first mined all frequent trees in a spanning tree forest from a general graph database and then retrieved all frequent maximal subgraphs from the trees.

Nijssen et al. in 2004 proposed a frequent substructure mining method called GASTON (GrAph/Sequence/Tree extractiON) [202]. GASTON is capable of mining frequent pathes, trees and graphs based on the pattern enumeration procedure of path-¿tree-¿graph. GASTON acquired the free trees from the same backbones (pathes) of existing tree free patterns and enumerated new frequent free tree candidates. It enumerated new graph pattern candidates by the cycle closing refinement operation that connects existing nodes in on a free tree or path.

Liu et al. in 2009 proposed a new subgraph pattern mining method called JPMiner [177]. JPMiner was designed to mine jump patterns alleviate the issue of the explosive number of patterns on the condition of low support value thresholds. A subgraph pattern is called σ-jump pattern if the differences between its and its super-graphs' support values are more than σ. JPMiner integrated the operation of checking jump patterns into the well-known DFS code tree enumeration framework.

Hsieh et al. in 2010 proposed a new closed subgraph mining algorithm TSP (Temporal

Subgraph Pattern algorithm) [121] to mine frequent temporal closed subgraph pattern from temporal heterogeneous information networks (HIN). In HIN, the nodes could be various types of entities and the edges represent multiple interactions between entities and time intervals on these interactions. TSP introduced the TSP-tree, k-level of which only exists k-length (the number of edges) patterns, to enumerate frequent patterns. It followed the extension enumeration strategy to do the DFS along the TSP tree.

Li et al. in 2011 proposed two novel graph mining algorithms RP-FP and RP-GD [174] to mine the representative pattern set to summarize the frequent subgraphs in a graph database. They used σ-jump patterns [177] as the representative patterns. RP-FP extracted a representative set from frequent closed subgraph patterns. It provided a tight ratio bound, but its computational requirement is intense. RP-GD used three heuristic operations, Last-Succeed-First-Check, Reverse-Path-Trace and Nephew-Representative-Based-Cover, to improve the computational cost. But it cannot provide a ratio bound.

2.5 Probabilistic Graphical Models

Probabilistic Graphical Models (PGM) are important unsupervised learning techniques that use a graph-based representation as the basis for compactly encoding a complex distribution over a high dimensional space [157]. PGM models are explored since late of 1980s to address problems of tabular probabilistic models, such as the exponential storage requirements, the exponential cost of computing marginal and conditional probabilities, and the lack of explicitness of the models [232].

In PGM models, nodes represent variables in various problem domains and edges represent the probabilistic relations between variables. They are capable of providing a compact and factorized form for the joint distribution of a set of variables.

The statistical graphical models mainly consist of two types of important families of models, Bayesian Networks (BNs) [116] and Markov Networks [255], based on the property of

graph edges (directed or undirected) [157]. In Bayesian Networks, the dependencies of distributions are encoded as acyclic directed graphs while in Markov Networks the dependencies are represented as undirected graphs.

2.5.1 Learning Markov Networks

In Markov network, an undirected graph structure $G : G = (X, E)$ with the nodes representing random variables $X = \{X_1, \ldots, X_n\}$ and the edges in the graph representing conditional independencies among X. Given all the maximum cliques C in G, $P(X) = \frac{1}{Z} \prod_{c_i \in C} g_{c_i}(X_{c_i})$ where $g_{c_i}(X_{c_i})$ is a non-negative potential function of X_{c_i}, X_{c_i} is the nodes of the clique c_i, and Z is the normalization constant that is equal to $\sum_X \prod_{c_i \in C} g_{c_i}(X_{c_i})$. $P(X)$ is called the Gibbs distribution.

There are three data analysis tasks in the Markov network problem: (I) Inference problem; (II) Parameter learning problem; (III) Structure learning problem.

For the task (I), given a graphical model G, the models need to calculate the marginal probability of a single variable or a set of variables. For the tasks (II) and (III), the models need to learn the parameters and graph structures from data.

For the inference problem, the major inference algorithms have variable elimination [280], belief propagation [275], message-passing [254, 265], and Power EP [189] etc.. For the parameter learning problem, the Maximum Likelihood Estimation (MLE) guarantees the global optimum although the closed form is impossible. The approximation and heuristic methods designed to reduce the computational cost of with iterative methods have simple gradient ascent [188], stochastic gradient [251], Score Matching [131], and the loopy belief propagation [274] etc.. For the structure learning problem, there exist two types of algorithms: score based approaches and independence based approaches. The score based approaches are used for the further inference tasks. The major methods have the top-down search that enumerates the candidates improve the conditional log-likelihood [186], L1-regularized based search [223], bottom-up search [66] and reference therein. The independence based approaches are

used to investigate the local structures in the networks. The methods have statistical independence tests [242, 181], local-to-global strategy [14], the Grow-Shrink Inference-based Markov Network (GSIMN) [32], the Particle Filter Markov Network (PFMN) algorithm [182], and reference therein. For a recent survey for markov network learning and inference, refer to [157, 232]

2.5.2 Learning Bayesian Networks

A Bayesian Network uses a directed acyclic graph to encode probabilistic dependency relationships among variables of interest [116]. It is another important probabilistic graphic model. A static BN is defined by a graph structure G, and a complete joint probability distributions of its nodes $P(X) = P(X_1, \ldots, X_n)$. The structure $G : G = (X, E)$ is an directed acyclic graph (DAG), which contains a set of variables $X = \{X_1, \ldots, X_n\}$, and a set of directed edges E, which define the causal relations between variables. Since the graph structures of static BNs are directed acyclic, the joint distributions can be decomposed as $P(X_1, \ldots, X_n) = \prod_i P(X_i|\pi_i)$, where π_i is the parents of the node (variable) X_i. We called this decomposition as the chain rule of conditional probability.

There are mainly three major problems in learning Bayesian Networks: (I) inference problem; (II) probability learning problem; (III) structure learning problem.

The inference task in Bayesian Network aims to calculate the posterior marginal probability or the most probable instantiation (the most probable explanation) of a single node or a set of nodes given the evidence E [211, 106]. It was proved that exact probabilistic inference in general is NP-hard [59]. The important exact inference algorithms include Pearl's work on message propagation inference algorithms [209, 210], junction tree algorithm [168], Arc reversal/node reduction algorithms [233, 234], and Variable elimination (VE) algorithm [280]. To alleviate the computational concerns of exact inference algorithms, researcher developed approximate algorithms that contain Monte Carlo algorithms [179, 47], Model Simplification algorithms [250, 147, 166], Search based algorithms [72, 230], and Loopy Belief Propagation

algorithms [259, 207].

The probability learning problem in Bayesian network aims to learn the parameters in a Bayesian network structure. In general there are two assumption settings in probability learning problem: the maximum likelihood learning (MLE) and the Bayesian setting [65]. In MLE the parameters are exact probabilities while in Bayesian the parameters are for the conditional density functions to model the conditional distributions in a network structure. The assumptions for data could be multi-nomial distributions [34, 115, 36] or normal distributions [87, 118, 191]. The probability learning problem usually is treated as part of learning the structure of a Bayesian network. The parameterizations are always indicated in the scoring of structures.

The Bayesian network structure learning problem is also NP-hard proven by Chickering [51]. A polynomial time complexity could be reached by learning structures with bounds that was shown in Ziegler's work [285]. The widely used scoring functions to evaluate the structures have Bayesian Dirichlet (BD) metric [115], Bayesian information criterion (BIC) metric [115], Akaike information criterion metric (AIC) [115], and Minimum Description Length (MDL) metric [28] . The most widely used algorithms for structure learning are heuristic scoring & searching algorithms and model averaging approaches. The heuristic searching algorithms include greedy search [60, 52], Genetic algorithms [167, 257], and Simulated annealing [67] etc.. The model averaging methods are usually applied into the applications with not much data and no models above others. The dominant techniques in model averaging are Markov Chain Monte Carlo (MCMC). The latest related work could be found in [257, 75] and reference therein.

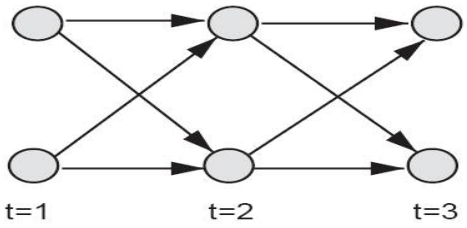


Figure 2.1: An example of dynamic bayesian network.

2.6 Learning Bayesian Network Structures from Stream Data

In many applications, data from which bayesian network models are learned are not simple sets and instead are temporal data streams. Hence, models that not only show the relationship between variables but also build the connections between the variable state trajectories over time may explain data better than the static bayesian networks.

In order to temporal processes in stream data, Dynamic Bayesian Network (DBN) was proposed to represent the temporal dependency among the data. It consisted of two components: initial network and temporal transition network [193]. In the DBN setting, the basic important assumption is the Markov assumption, that is, that $P(X^t|X^{0:(t-1)}) = P(X^t|X^{t-1})$. X^t are the observations at time t. An example of dynamic bayesian network structure is shown in Figure 2.1.

The early related work on DBN models could be found in [193, 155, 129, 278]. Recently, considering that the underlying stochastic processes that generate the time series expression data may not be stationary, non-stationary DBN models have attracted significant research interests [227, 102, 240, 132, 170, 70, 130, 103].

The change-point detection problems have been extensively investigated in time series models. Recent work could be found in: PCA-based singular-spectrum transformation models [192]; non-parametric online-style algorithm via direct density-ratio estimation [153]; two-phase linear regression model [178]; a hybrid algorithm that relies on particle filtering and Markov chain Monte Carlo [54]; the RJMCMC method [100]; The perfect simulation

model based on product-partition [77]; change-point detection by minimizing a penalized contrast function [169]. Researchers found that change-point modeling is a very promising way of dealing with the non-stationarity property [54]. Hence, the current non-stationary DBNs methods employed different change-point detection techniques to model the underlying change-point processes of network structures.

Based on different strategies of applying these change-point detection techniques, we divided the existing non-stationary DBN methods into two categories: purely change-point detection based approaches [227, 102, 103] and constantly varying network learning approaches [240, 132, 170, 70, 130]. Change-point detection approaches aim to learn a sequence of structures and the posterior distribution of the change-points of these structures while time varying network learning approaches are designed to learning structures for each time points.

2.6.1 Change-point based Non-stationary Dynamic Bayesian Networks

Robinson et al. in 2008 [227] applied Reversible Jump Markov Chain Monte Carlo (RJMCMC) [100] to model the non-stationary Bayesian Network problem. They designed 10 move types to span the model space. They provided 3 settings that are Unknown Number of transitions and Unknown Time of transitions (UNUT), Known Number of transitions and Unknown Time of transitions (KNUT), and Known Number of transitions and Known Time of transitions (KNKT). They used a discrete model with the assumed multinomial distributed data with the Dirichlet prior for observations and derived an extended BDeu score to evaluate the quality of structures.

Grzegorczy et al. in 2008 [102] applied the allocation sampler technique and introduced a continuous-valued DBNs method that approximates the non-stationary property with a Gaussian mixture model. They assumed Gaussian distribution with the normal-Wishart prior for the time series observations. They used the Dirichlet prior for the weights of

mixture vector and the truncated Poisson prior for the number of mixture labels. They designed six Markov Chain Monte Carlo (MCMC) move types to to traverse the model state space.

Later in 2011 Grzegorczy et al. [103] used perfect simulation technique to improve the convergence of their sampling approach based on the same allocation sampler model [102]. They used the Metropolis Hasting algorithm to approximate the structure state space and used perfect simulation to simulate the mixture vector of change points at each step of sampling step. They also followed the work in [170] to decompose the complete variable set into small clusters that reduced the computational cost.

Noticing the low estimation accuracy of Robinson's work based on RJMCMC [227] and potential computational concern of Grzegorczy's approach [103], Jia et al. in 2012 [145] proposed a new method that used the MCMC approach to sample the model state and adapted the perfect simulation model to our multi-variate time series data. The new model integrated with the dynamic bayesian network modeling. The algorithm is designed in an iterative way that only directly simulate the change-points of structures at the end of each iteration. The variances of structures between iterations are monitored to decide the convergence of the output.

Based on their work, Ickstadt et al. [132] further generalized this non-linear BGe mixture model into a broader framework of non-parametric Gaussian Bayesian networks.

2.6.2 Time Varying Non-stationary Dynamic Bayesian Networks

Song et al. in 2009 [240] proposed a time varying dynamic bayesian network method to learn structures for each time point in the time series data. They used kernel re-weighting functions to aggregate the information across time points. They used l1-regularized auto-regressive least square solver to learn the structures. The optimization problem of the objective function was solved by the shooting algorithm [85].

Lebre et al. in 2010 [170] proposed a new time varying networks approach based on

first-order auto-regression and Yao's two-stage regime-SSM model [222]. They used the RJMCMC technique to sample the structures. Their method focuses on local structural changes and performs node-by-node analysis. They used l1-regularized regression model with the assumption of Gaussian distributions with the zero means on coefficient parameters.

In order to address the structural overfitting problem in [170], Dondelinger et al. in 2010 [70] introduces information sharing between segments into Lebre's approach by introducing a regularization scheme based in inter-time segment information sharing. They used two information sharing strategies that include sequential information sharing based on the work of [261] and global information sharing based on the work of [79].

Husmeier et al. in 2010 [130] investigated three regularization schemes based on inter-segment information sharing to reduce the risk of overfitting and inflated inference uncertainty. The first scheme was proposed by using the hard information coupling based on the work on [261]. The second scheme used the hard information coupling by applying a binomial distribution with conjugate Beta prior. The third one used a soft information coupling strategy by applying a binomial distribution with conjugate Beta prior.

Chapter 3

Preliminary Work (I): Approximate Graph Mining based on Evolutionary Process

We designed an approximate graph mining algorithm based on evolutionary process [139, 140, 143]. We applied our algorithm to both synthetic and real data sets. The experimental results demonstrate that our algorithm identifies important subgraphs that can not be identified by exact matching algorithms with a pattern discovery speed (number of patterns divided by the running time) close to, and sometime better than, conventional exact matching algorithms.

3.1 Introduction

Frequent subgraph mining is an active research topic in the data mining community. The graph mining techniques have been extensively applied in a wide range of applications domains, such as bioinformatics [122, 126], chemoinformatics [119, 217], social network analysis [164, 269], and many others.

Many current frequent subgraph mining algorithms share a common strategy in determining the support value of a subgraph pattern and hence deciding whether the subgraph is

frequent. In this strategy, in matching a subgraph pattern to a graph, we require that node labels, edge relationships, and edge labels should be the same between the subgraph pattern and the matching graph [123]. We call this strategy *exact matching* [1].

Although exact matching is widely used, in applying frequent subgraph mining to real-world applications, we observe that exact matching may not always produce the optimal results in all applications. The situation becomes worse in those graph databases that have a large volume of noises (in terms of node or edge label changes) and distortions (in terms of edge relationship changes). For example, in the application of protein structure comparison and structure motif identification, which we are specifically interested in within this paper, graphs corresponding to protein structures often contain a large volume of noises and distortions. In this application, noise and distortion come from a multidimensional source: amino acid changes in proteins (which are called *mutations* in biology) , slightly different geometric shape of similar proteins, and imperfect experimental measurements, just to name a few examples. As a consequence, using exact matching posts an unrealistical constraint in algorithm design and may miss a lot of important patterns in practice.

The goal of our research is to devise frequent subgraph mining algorithms that are capable of identifying salient patterns in large graph database that are otherwise overlooked by using exact matching due to the presence of noises and distortions in the graph databases. We call this new strategy *approximate graph mining*.

We designed a new approximate subgraph mining method called APGM (APproximate Graph Mining). We developed a general framework that uses a probability matrix to score label mismatches in matching a subgraph pattern to a graph. The advantage of the strategy is that it holds a solid probabilistic ground for a whole class of applications. Utilizing this scoring scheme, we renewed important definitions, such as isomorphism, subgraph isomorphism, and redesigned the conventional support measures in this new context. We designed

[1]Technically, we should use *subgraph isomorphism* to define exact matching. The definition of subgraph isomorphism is deliberately delayed to a later section. An intuitive description is provided here to avoid excessive details in the introduction

a depth-first search strategy with a set of pruning strategies.

3.2 Related work

Graph database mining is an active research field in data mining research. The goal of graph database mining is to locate useful and interpretable patterns in a large volume of graph data. Current exact matching graph mining algorithms can be roughly divided into three categories. The first category uses a level-wise search strategy including AGM (Apriori based Graph Mining) [135] and FSG (Frequent Subgraphs) [160]. The second category takes a depth-first search strategy including gSpan (Graph based Substructure PAtterNmining) [270] and FFSM (Fast Frequent Subgraph Mining) [124]. The third category works by mining frequent trees, in which SPIN (SPanning tree based maximal graph mINing) [128] and GASTON (GrAph/Sequence/Tree extractiON) [202] are the representative. Recently, researchers extend the graph mining problem from static networks into temporal dynamic networks [163] or involving networks [38]. We refer to [112] for a recent survey.

Frequent subgraph mining with approximate matching capability has also been investigated. Chen et al. proposed a method called gapprox [43], which discovers approximate matched patterns in a single large network. Yan et al. designed a graph query algorithm Grafil (Graph Similarity Filtering) for approximate structure data search [273]. The algorithm SUBDUE [119] considers the situation of inexact matching and includes a distortion cost function as a solution. Zhang et al. provided a method called Monkey [282] to identify maximal approximately frequent trees. Further the same group introduced a randomized algorithm called RAM to find approximate subsequent subgraphs by using feature retrieval to avoid canonical form calculation [281]. Zou et al. proposed an approximation algorithm MUSE (Mining Uncertain Subgraph pattErns) focusing on uncertain graph database [287]. This method calculated the expected support values of patterns by considering both the occurrences in the uncertain graph databases and the probabilities of the uncertain graph

databases.

The differences between existing algorithms and our proposed one are below. Yan's work focuses on proximity measures between graphs and Chen's work concentrates on pattern discovery in a single large graph, which are out of the scope of our current paper. SUBDUE did not provide a complete general frame to address the approximate match issue. It is only applied to small databases and generates an incomplete set of characteristic subgraphs. By using a feature set instead of the canonical form to distinguish patterns, RAM may not provide a complete pattern set. Hence, the algorithm's efficacy highly depends on the quality of user-defined feature set. Different from our method and other methods, instead of the deterministic data, MUSE addressed the uncertain data with inherent statistical properties in nature [12, 13, 260]. It only handles the uncertain edges and quantifies the uncertainty with the probability distributions.

Different from these existing works, we use a parametric model to determine the probability that a pattern belongs to a graph. We developed a general framework to fully utilize a probability matrix for approximate matching, which we can apply to a number of different applications. And our theoretic framework promises the completeness of the pattern set. Finally we offered a practical implementation, applied it on both synthetic and real data sets, and evaluated our method rigorously.

3.3 Theoretic Framework

We demonstrate our method called **APGM** (**AP**proximate **G**raph **M**ining) with two steps. In this section, we introduce the theoretic model. In the next section, we show our algorithm in details.

Definition 1. *A **labeled graph** G is a 5-tuple $G = \{V, E, \Sigma_V, \Sigma_E, \lambda\}$ where V is the set of vertices of G and $E \subseteq V \times V$ is the set of undirected edges of G. Σ_V and Σ_E are (disjoint) sets of labels and labeling function $\lambda : V \to \Sigma_V \cup E \to \Sigma_E$ maps vertices and edges in G to*

their labels. A ***graph database*** *D is a set of graphs.*

We also use $V[G]$ to denote the node set of a graph G and $E[G]$ to denote the edge set of G. We also use $\Sigma_{V[G]}$ to denote the node labels, $\Sigma_{E[G]}$ to denote edge labels, and λ_G to denote the labeling function for a graph G. Before we introduce approximate matching, we define the exact subgraph isomorphic and the compatibility matrix.

Definition 2. *A graph G is* ***subgraph isomorphic*** *to another graph G', denoted by $G \subseteq G'$ if there exists an injection $f : V \to V'$, such that*

- $\forall\ u \in V, \lambda(u) = \lambda'(f(u))$
- $\forall\ u, v \in V, (u, v) \in E \Rightarrow (f(u), f(v)) \in E'$, *and*
- $\forall\ (u, v) \in E, \lambda(u, v) = \lambda(f(u), f(v))$

Definition 3. *A* ***compatibility matrix*** *$M = (m_{i,j})$ is an $n \times n$ matrix indexed by symbols from a label set Σ ($n = |\Sigma|$). An entry $m_{i,j}$ ($0 \leq m_{i,j} \leq 1$, $\sum_j m_{i,j} = 1$) in M is the probability that the label i is replaced by the label j.*

The compatibility matrix offers a probability framework for approximate subgraph mining. A compatibility matrix M is *stable* if the diagonal entry is the largest one in the row (i.e. $M_{i,i} > M_{i,j}$, for all $j \neq i$). In a stable compatibility matrix, for any label i it is likely to be replaced by itself rather than by any other symbols.

Most compatibility matrices in real-world applications are stable or almost-stable ones, and hence for the rest of this section, we only deal with the stable compatibility matrices.

Definition 4. *A graph G is* ***approximate subgraph isomorphic*** *to another graph G', denoted by $G \subseteq_a G'$ if there exists an injection $f: V \to V'$, such that*

- $\prod_{u \in V} \frac{M_{\lambda(u),\lambda'(f(u))}}{M_{\lambda(u),\lambda(u)}} \geq \tau$,
- $\forall\ u, v \in V, (u, v) \in E \Rightarrow (f(u), f(v)) \in E'$, *and*

- $\forall\ (u,v) \in E, \lambda(u,v) = \lambda(f(u), f(v))$

Given a node injection f from graph G to G', the co-domain of f is an *embedding* of G in G'. The *approximate subgraph isomorphism score* of f, denoted by $S_f(G,G')$, is the product of normalized probabilities: $S_f(G,G') = \prod \frac{M_{\lambda(u),\lambda'(f(u))}}{M_{\lambda(u),\lambda(u)}}$. For a pair of graphs, there may be many different ways of mapping nodes from one graph to another and hence may have different approximate isomorphism scores. The *approximate matching score* (score for simplicity) between two graphs, denoted by $S(G,G')$, is the largest approximate subgraph isomorphism score, or

$$S(G,G') = \max_f \{S_f(G,G')\}$$

.

Definition 5. *Given a graph database D, an isomorphism threshold τ, a support threshold σ $(0 < \sigma \leq 1)$, the* ***support value*** *of a graph G, denoted by sup_G, is the average score of the graph to graphs in the database, which G is approximately subgraph isomorphic to:*

$$sup_G = \sum_{G' \in D, G \subseteq_a G'} S(G,G')/|D| \tag{3.1}$$

G is a *frequent approximate subgraph* if its support value is at least σ. With this definition, we only use those graphs that a subgraph G is approximate subgraph isomorphic to (controlled by the parameter τ) to compute the support value of G. We do this to filter out low quality (but potentially many) graph matchings in counting the support value of a subgraph. For a moderate sized graph database $(100 - 1000)$, according our experience, the number of frequent subgraphs identified is usually not sensitive to the isomorphism threshold, which makes sense since low quality graph matching has low "weight" in the support computation nevertheless.

With the above definition, we have the support Apriori property as claimed by the following Theorem 1.

Theorem 1. *Given a graph database D and two graphs $G \subseteq G'$, we have $sup(G) \geq sup(G')$.*

Proof. In order to prove the theorem, it is sufficient to show that for all graphs P in a graph database, we have $S(G, P) \geq S(G', P)$ for all graphs $G \subseteq G'$. This is true if the compatibility matrix is stable ($m_{i,i} > m_{i,j}$ for all $j \neq i$). The rest of the proof are trivial and are left to interested readers. □

Problem Statement. Given a graph database D, an isomorphism threshold τ, a compatibility matrix M, and a support threshold σ, the *approximate subgraph mining* problem is to find all the frequent approximate subgraphs in D.

It is easy to adapt the frequent approximate subgraph mining algorithm to the approximate clique subgraph mining by adding the full-connection constraint. In order to keep the consistency with our real world applications, The subgraphs shown in all the examples below are clique subgraphs instead of subgraphs.

Theorem 2. *Given a graph database D, an isomorphism threshold $\tau = 1$, a compatibility matrix M, and a support threshold σ, the set of approximate frequent subgraph patterns P_a is exactly the set of frequent subgraph patterns P_f or $P_a = P_f$. If $\tau < 1$, $P_f \subseteq P_a$.*

Proof. This is the direct consequence of the support value definition 5. □

3.4 Algorithm Design

Here we demonstrate a new algorithm APGM for approximate subgraph mining. APGM starts with frequent single node subgraphs. At a subsequent step, it adds a node to an existing pattern to create new subgraph patterns and identify their support value. If none of the resulting subgraphs are frequent, APGM backtracks. APGM stops when no more patterns need to be searched. Before we proceed to the algorithmic details, we introduce the following definitions to facilitate the demonstration of the APGM algorithm.

Definition 6. *Given a graph T, one of the embeddings $e = v_1, v_2, \cdots, v_k$ of T in a graph G, a node v is a* ***neighbor*** *of e if $\exists u \in e, (u, v) \in E[G]$.*

In other words, a neighbor node of a embedding e is any node that connects to at least one node in e. The *neighbor set* of an embedding e, denoted by $N(e)$, is the set of e's neighbors.

Definition 7. *Given a graph T, one of the embeddings $e = (v_1, v_2, \cdots, v_k)$ of T in a graph G, an injection f from T to e, an isomorphism threshold τ, a compatibility matrix M, a node $v \in N(e)$, and a node label l, the* ***approximate subgraph pattern candidate****, denoted by $G|_{T,e,v,l}$, is a graph $(V', E', \Sigma_{V'}, \Sigma_{E'}, \lambda')$ such that*

- $\lambda'(v) = l$
- $V' = \{v_1, v_2, \cdots, v_k\} \cup v$
- $E' \subseteq V' \times V' \cap E[G]$
- $\Sigma_{V'} = \Sigma_{V[T]}$
- $\Sigma_{E'} = \Sigma_{E[T]}$
- $\forall\ u, v \in V' : \lambda'((u, v)) = \lambda_G(f(u, v))$
- $\prod_{u \in V'} \frac{M_{\lambda'(u), \lambda_G(f(u))}}{M_{\lambda'(u), \lambda'(u)}} \geq \tau$

With the the definitions, we present the pseudo code of APGM below.

APGM_MAIN(D, M, τ, σ)

1: Begin
2: $C \leftarrow$ { frequent single node }
3: $F \leftarrow C$
4: **for** each $T \in C$ **do**
5: $\quad APGM\_SEARCH(T, \tau, \sigma, F)$
6: **end for**
7: return F
8: End

APGM_SEARCH(T, τ, σ, F)

```
1: Begin
2: C ← ∅
3: for each (e, v), e is an embedding of T in a graph G, v ∈ N(e) do
4:    CL ← approximateLabelSet(T, G, e, v)
5:    for each l ∈ CL do
6:       X ← G|_{T,e,v,l}
7:       C ← C ∪ {X}
8:       H(X) = H(X) ∪ (e, v)
9:    end for
10: end for
11: remove infrequent T from C
12: F ← F ∪ C
13: for each T ∈ C do
14:    APGM_SEARCH(T, τ, σ, F)
15: end for
16: End
```

$\mathcal{H}$ is a hash function to store candidate subgraphs and their embeddings. The hash key of the function in our implementation is a canonical code of the subgraph X, which is a unique string representation of a graph. We use the Canonical Adjacency matrix (CAM) and the Canonical Adjacency Matrix code, developed in [123], to compute the canonical code of a graph.

approximateLabelSet(T, G, e, v)

```
1: Begin
2: R ← ∅
3: l_0 ← λ_G(v)
4: for each l ∈ Σ_{V[G]} do
5:    if S(e, T) × M(l_0,l)/M(l_0,l_0) ≥ τ then
6:       R ← R ∪ l
7:    end if
8: end for
9: return R
10: End
```

APGM enumerates the subgraph candidates from the new proposed embeddings. The procedure to find new embeddings are described in Definition 7. The information of neighbors are collected at the beginning of Algorithm APGM_MAIN. When all the new embedding are

enumerated based on the embeddings of an existing subgraph, APGM has the new subgraph candidates and each candidate has all its embeddings. The support value of a new subgraph candidate is calculated by following Definition 4, 5 and 6. We gave one example below to show the enumeration procedure of patterns and their embeddings.

3.5 Experimental Study

3.5.1 Data Sets

We downloaded all protein structures from Protein Data Bank (PDB). We followed [19] to use the same software as [127] to calculate Almost-Delaunay (AD) for graph representation of protein geometry. We took BLOSUM62 [205] as the compatibility matrix and back-calculated the conditional probability matrix by following the procedure described in [76]. In the BLOSUM62 substitution matrix, there is only one violation of the criterion of stable matrices– the row for methionine (MET). We normalized the row of MET with the maximum entry inside it and other rows in the matrix according to $Definition$ 4. We investigated two immunologically relevant protein domain families: the Immunoglobulin V set and the Immunoglobulin C1 set. Immunoglobulin domains are among the most valuable to give insight into host-defense mechanisms, and insight that can help guide development of therapies and vaccines against refractory organisms[149]. We collected proteins from SCOP release 1.69. For each family we created a culled set of proteins with maximal pairwise sequence identity percentage below 30% by using PISCES server[256]. The PDB ID of Individual proteins for two sets are shown in $Table$ 3.1. The graph properties of two protein families are listed in $Table$ 3.2. We denote Immunoglobulin domain proteins as positive sample and random proteins as negative.

Table 3.1: Immunoevasins Protein Lists for Research

	PDB ID of proteins in Immunoglobulin C1 set
Proteins for Feature Extraction (10):	1*fp*5*a* 1*onqa* 1*ogad* 1*pqza* 1*t*7*va* 1*l*6*xa* 1*je*6*a* 1*mjul* 1*uvqb* 1*dn*0*b*
Proteins for Leave-one-out Testing (11):	1*nfda* 1*uvqa* 1*q*0*xl* 1*mjuh* 1*a*6*za* 1*k*5*na* 1*hdma* 3*frua* 1*ogae* 1*hdmb* 1*k*5*nb*
	PDB ID of proteins in Immunoglobulin V set
Proteins for Feature Extraction (10):	1*pkoa* 1*ogad* 1*npua* 1*cdca* 1*jmaa* 1*fo*0*b* 1*nkoa* 1*mjuh* 1*nfdb* 1*qfoa*
Proteins for Leave-one-out Testing (9):	1*zcza* 1*f*97*a* 1*eaja* 1*mjul* 1*cida* 1*neua* 1*cdya* 1*hkfa* 1*nezg*

Table 3.2: Graph Properties of Immunoevasins Proteins

	Immunoglobulin C1 set	Immunoglobulin V set
avg. node size	220	158
avg. edge size	3107	2263
max. node size	276	159
min. node size	100	96
max. edge size	4000	4030
min. edge size	1350	713
avg. density	14	14
node label size	20	20
edge label size	27	30

3.5.2 Results

During this experimental research, we mined frequent clique subgraphs[125] in order to enforce biological constraints on the patterns. We compared APGM with the exact graph mining methods MGM. We chose MGM as the counterpart for the comparison because it is an available clique pattern mining algorithm. (Any exact matching method with clique constraint should provide the same number of patterns from a graph database.)

Experimental Protocol. We created our experimental protocol as the following:

- We randomly chose 10 proteins from each family as group I to serve as sources for feature extraction.
- We used the remainder (positive sample) as group II for training and testing in "leave-one-out" cross validation.
- A negative sample set of the the same size as the positive samples in group II was randomly chosen from PDB. The negative sample was used along with the positive sample in training and testing.

The complete flowchart of our experiment procedure is shown in *Fig.* 3.1.

- In order to eliminate the effect of randomness on our classification results, we chose the optimal parameters to repeat the procedure shown in *Fig.* 3.1 100 times for each data set.

Number of Patterns Identified. We identified frequent approximate subgraph patterns from 10 positive proteins in each family. There are two parameters that may have significant influence on the set of mined patterns. The first is the support threshold (σ) and the second is the isomorphism threshold (τ). For simplicity, in the following experiments in this section we use the new support threshold $\sigma' = \sigma \times |D|$, where $|D|$ is the size of the graph database,

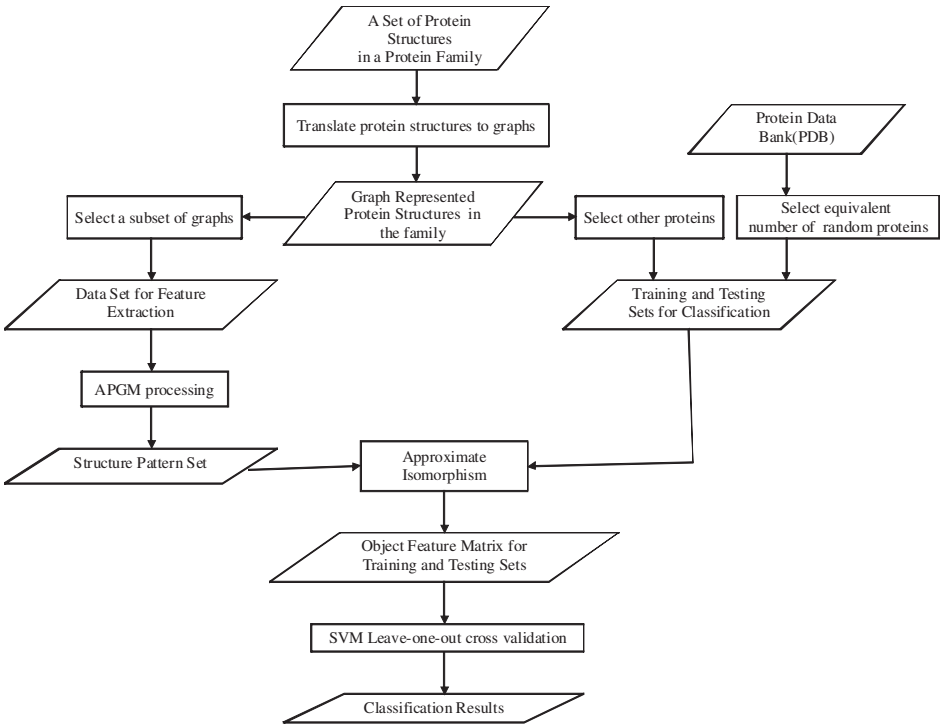


Figure 3.1: The procedure of experimental research

and applied the same change in support value. In *Table* 3.3, we run APGM with different combinations of τ and σ and collect the total number of identified patterns. Our results show that the total number of patterns is not sensitive to the isomorphism threshold, and depends on the support threshold heavily. Such fact eases the worry that the parameter τ may be too strong for deciding the number of patterns.

For the purpose of comparison, the patterns mined by two mining methods are shown in *Table* 3.4 and 3.5, and the patterns acquired by APGM from Immunoglobulin C1 proteins are also shown in *Table* 3.3. In our experiment, we treat a pattern set with the number more than $10,000$ as a meaningless one because our sample space is comparatively small and the isomorphism check is computationally expensive. From *Table* 3.5, we see that exact matching fails to provide useful patterns on the Immunoglobulin V proteins, which is the typical data set with very noisy background. In comparison, APGM does find some pattern set with a reasonable size in such situation. (We only use rough parameter combination grids to do the pattern search. If we increase the precision of τ and σ , more patterns will

Table 3.3: Number of Patterns for Immunoglobulin C1 Set acquired by APGM.

	$\tau = 3.5$	$\tau = 4.5$	$\tau = 5.5$
$\sigma = 4$	811	774	750
$\sigma = 5$	141	140	136
$\sigma = 6$	17	17	17

Table 3.4: Number of Patterns by APGM ($\tau = 0.35$) and MGM on Immunoglobulin C1

	Support Threshold (σ)				
	6	5.5	5	4.5	4
$APGM(\tau = 0.35)$	17	24	141	202	841
MGM	16	16	126	126	660

Table 3.5: Number of Patterns by APGM ($\tau = 0.75$) and MGM on Immunoglobulin V

	Support Threshold (σ)				
	6	5.5	5	4.5	4
$APGM(\tau = 0.75)$	0	0	0	160	14686
MGM	0	0	0	0	13911

be found.) In order to evaluate the quality of these patterns, we use the identified frequent subgraphs in classification tests as discussed below.

Classification Performance. In this experimental section, we used *libsvm* SVM package (`http://www.csie.ntu.edu.tw/thicksimcjlin/libsvm`) for protein structure classification. We treat each mined pattern as a feature and a protein is represented as a feature vector $V = (v_i)$ where $1 \leq i \leq n$ and n is the total number of identified features. v_i is 1, if the related feature occurs in the protein and otherwise v_i is 0. We used the linear kernel and default parameters for SVM leave-one-out cross validation, where $Accuracy = (TN + TP)/(TN + TP + FN + FP)$ (TP, true positive; FP, false positive; TN, true negative; FN, false negative).

We followed the procedure in *Fig.* 3.1 to create one data set for feature extraction and another for training and testing on both Immunoglobulin C1 and V proteins. The classification results are summarized in *Table* 3.6 and 3.7. For some parameter combinations, there are no accuracies - an event which happens under two circumstances. First, there are no patterns found. Second, the pattern set is too big to be useful. From the tables we see

Table 3.6: Classification Accuracy of APGM ($\tau = 0.35$) and MGM on Immunoglobulin C1 Set

	Support Threshold (σ)				
	6	5.5	5	4.5	4
APGM	68.18%	77.27%	86.36%	90.91%	81.82%
MGM	72.73%	72.73%	72.73%	72.73%	72.73%

Table 3.7: Classification Accuracy of APGM $\tau = 0.75$)and MGM on Immunoglobulin V Set

	Support Threshold (σ)			
	6	5.5	5	4.5
APGM	–	–	–	77.78%
MGM	–	–	–	–

TP, true positive; FP, false positive; TN, true negative;
FN, false negative.
Accuracy = (TN+TP)/(TN+TP+FN+FP).
– means accuracies are unavailable.

that the classification with APGM-based feature highly outperforms those based on exact matching. For Immunoglobulin C1 set, the classification based on feature identified by MGM only reaches 73%, while APGM is between $69\% - 91\%$. For Immunoglobulin V set, since the exact matching method cannot mine any meaningful patterns, it fails in classification, while by using APGM, we have the accuracy about 78%. It shows that our APGM has more capability to mine useful structure information from very noisy background than general exact matching graph mining algorithms.

We repeated the experimental procedure 100 times for both protein families. We showed the results of average Accuracy and its variance in *Fig.* 3.2 and 3.3, and the results of average Precision and Recall and their variance in *Table* 3.6 and 3.7. In all of three classification measures, APGM outperformed the exact matching method MGM, which demonstrates our previous finding in the previous single experiment.

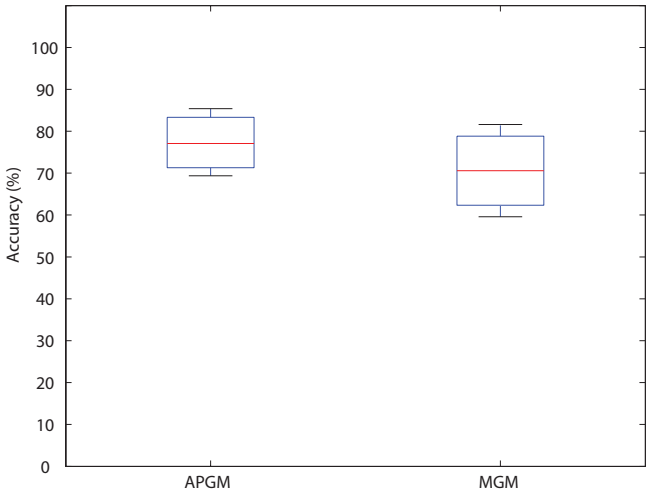


Figure 3.2: The Accuracy comparison between APGM and MGM on Immunoglobulin C1 set

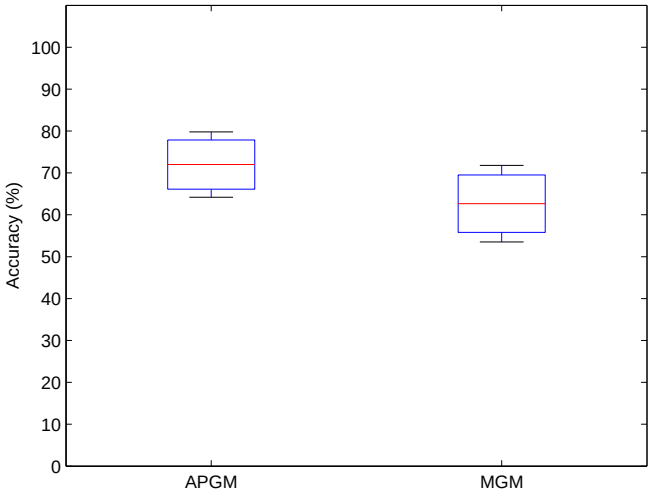


Figure 3.3: The Accuracy comparison between APGM and MGM on Immunoglobulin V set

Table 3.8: Prediction Comparison betwen APGM ($\tau = 0.35$) and MGM on Immunoglobulin C1

	$Precision\ (avg. \pm variance)$	$Recall\ (avg. \pm variance)$
$APGM$	$87.87 \pm 7.96\%$	$62.50 \pm 12.40\%$
MGM	$86.79 \pm 13.35\%$	$48.21 \pm 16.05\%$

Table 3.9: Prediction Comparison betwen APGM ($\tau = 0.75$) and MGM on Immunoglobulin V

	$Precision\ (avg. \pm variance)$	$Recall\ (avg. \pm variance)$
$APGM$	$92.90 \pm 11.63\%$	$47.57 \pm 13.24\%$
MGM	$86.26 \pm 17.72\%$	$30.53 \pm 13.67\%$

Precision = TP/(TP+FP).
Recall = TP/(TP+FN).
For the C1 set, APGM chose two optimal parameter combinations ($\tau = 0.35, \sigma = 4.5$) and ($\tau = 0.35, \sigma = 5$), and MGM chose two optimal parameters $\sigma = 5, 6$. In 200 mining times, APGM found 200 non-empty pattern sets and MGM found 185. For the V set, APGM chose two optimal parameter combinations ($\tau = 0.75, \sigma = 4.5$) and ($\tau = 0.75, \sigma = 5$), and MGM chose two optimal parameters $\sigma = 5, 6$. In 200 mining times, APGM found 192 non-empty pattern sets and MGM found 135.

Chapter 4

Preliminary Work (II): Dynamic Bayesian Networks based on RJMCMC

We proposed a novel non-stationary DBNs method [141]. Our method is based on Reversible Jump Markov Chain Monte Carlo (RJMCMC) [100] with a potential regulator detection technique and a flexible lag choosing mechanism. We apply the approach for the gene regulatory network inference on three non-stationary time series data. For the Macrophages and Arabidopsis data sets with the reference networks, our method shows better network structure prediction accuracy. For the Drosophila data set, our approach converges faster and shows a better prediction accuracy on transition times. In addition, our reconstructed regulatory networks on the Drosophila data not only share a lot of similarities with the predictions of the work of other researchers but also provide many new structural information for further investigation.

4.1 Introduction

Recently non-stationary Bayesian network models have attracted significant research interests in modeling gene expression data. In non-stationary Bayesian networks, we assume that the underlying stochastic process that generates the gene expression data may change over time. Non-stationary Bayesian networks have advantage over conventional methods in applications where the intrinsic regulatory networks are subject to changes for adapting to internal or external stimuli. For example, gene expression profiles may go through dramatic changes in different development stages [227], or in the invasion process of viruses [102], or as response to changes of outside environment such as temperature and light intensity [183].

Recent work on non-stationary Bayesian networks could be found in [227, 102]. Robinson's method [227] used RJMCMC (Reversible Jump Markov Chain Monte Carlo) to sample underlying changing network structures, in which an extended BDe metric (Bayesian-Drichlet equivalent) is applied. And Grzegorczy et al. [102] proposed a non-homogeneous Bayesian network method to model non-stationary gene regulatory processes, in which they included a Gaussian mixture model based on allocation sampler technique [204], provided an extended non-linear BGe (Bayesian Gaussian likelihood equivalent) metric and employed MCMC (Markov Chain Monte Carlo) to collect samples.

There are several limitations on the existing non-stationary DBNs methods that are discussed above. First, the RJMCMC that is used in Robinson's work [227] is a computationally expensive approach especially in dealing with gene networks. Second, mixture model used by Grzegorczy et al. avoided intensive computational issue by using MCMC, but it does not capture the underlying changing network structures over time. In addition, both methods used a fixed time delay $\tau = 1$ that leads to a relatively low accuracy of prediction on network re-construction [286].

In this paper, we proposed a new non-stationary DBNs approach extending the work presented in [227] and [286]. Our method modified RJMCMC by employing a systematic approach to determine potential regulators. We designed a flexible lag determine mechanism

by considering the delay in the gene expression changes between potential regulators and target genes. In this approach we efficiently reduce the model searching space, capture the dynamics of transcriptional time delay, and speed up computation with a fast convergence.

4.2 Related Work

With a well-defined probabilistic semantics and the capability to handle hidden variables [185], Dynamic Bayesian Networks (DBNs) are widely used on regulatory network structure inference from noisy microarray gene expression data [84, 193, 129, 113, 278, 133, 134, 155, 195, 24].

The early work of applying BNs to analyzing expression data could be found in [84, 193]. Many works have been done since then. Hartemink et al. extended the static BNs by including latent variables and annotated edges, and their work focused on scoring the models of regulatory network [113]. Considering the problem of information loss incurred by discretization of expression data, Imoto et al. proposed a continuous BNs and non-parametric regression model [133]. They used Laplace approximation to the marginal probability to infer a BNRC score as the scoring metric for network models. Further, Hartemink and Imoto extended their techniques to DBNs [155, 278]. Before the BNs, previous efforts at modeling genetic regulatory networks fell into two categories [113, 129]: fine-scale methods utilizing differential equations, and coarse-scale methods using clustering and boolean network models. BNs method is perceived as a good compromise of the two levels. With the challenging of small number of samples, researchers seek additional information such as transcriptional localization data[24], DNA sequences of promoter elements [134], and protein-protein interaction data[195] to improve the accuracy of gene networks reconstruction.

4.3 Method

Structure Learning of Non-stationary Bayesian Networks

Bayesian networks (BNs) are a special case of probabilistic graphic models. A static BN is defined by an acyclic directed graph G and a complete joint probability distribution of its nodes $P(X) = P(X_1, \ldots, X_n)$. The graph $G : G = \{X, E\}$ contains a set of variables $X = \{X_1, \ldots, X_n\}$, and a set of directed edges E, defining the causal relations between variables. With a directed acyclic graph, the joint distribution of random variables $X = \{X_1, \ldots, X_n\}$ are decomposed as $P(X_1, \ldots, X_n) = \prod_i P(X_i|\pi_i)$, where π_i are the parents of the node (variable) X_i.

The topology of bayesian networks must be a directed acyclic graph and hence could not be used to model the case where two genes may be a regulator of each other. As an extension of BNs to model time series data, Dynamic Bayesian Networks (DBNs) lift the limitation of directed acyclic graph by incorporating time in constructing bayesian networks. Given an observed time series data D spanning T time points, the structure learning problem of DBNs is equal to maximizing the posterior probability of the network structure G. By the Bayes' rule, the posterior probability is expressed as the following:

$$P(G|D, T) = \frac{P(D|G, T)P(G|T)}{P(D|T)} \tag{4.1}$$

The current application of DBNs to gene expression data assumes that the underlying stochastic process generating the data is stationary. Here we provide a new approach to capture the structural dynamics of non-stationary data.

We assume the time series gene expression profile is subdivided to m segments. In each segment, there is one graph $G_i : 1 \leq i \leq m$ that dominates the segment. Given a sequence of network structures $G^T = (G_1, \ldots, G_m)$, the posterior probability in Equation 1 is replaced

by Equation 2.

$$P(G^T, m|D, T) = \frac{P(D|G^T, m, T)P(G^T, m|T)}{P(D|T)} \tag{4.2}$$

In applying DBNs to gene expression data, we first decide the time lag value τ, which is the time delay between causes and effects in the time series data. Most previous work set $\tau = 1$ for modeling a first-order markov chain. However, evidence shows that higher-order markov chain might better model gene expression data and biological networks [286]. Given a maximum lag value τ_{max}, in corresponding to the graph structure sequence G^T, we assign a lag vector $\tau^T = (\tau_1, \ldots, \tau_m)$, in which $\tau_i : 1 \leq \tau_i \leq \tau_{max}$. So Equation 2 further extends to:

$$P(G^T, m, \tau^T, \tau_{max}|D, T) = \frac{P(D|G^T, m, \tau^T, \tau_{max}, T)P(G^T, m, \tau^T, \tau_{max}|T)}{P(D|T)} \tag{4.3}$$

$P(D|T)$ is treated as a constant, and then

$$\begin{aligned} P(G^T, m, \tau^T, \tau_{max}|D, T) &\propto P(D|G^T, m, \tau^T, \tau_{max}, T)P(G^T, m, \tau^T, \tau_{max}|T) \\ &\propto P(D|G^T, m, \tau^T, \tau_{max}, T)P(G^T|m, T) \\ &\quad P(\tau^T|m, \tau_{max}, T)P(m|T)P(\tau_{max}|T) \end{aligned} \tag{4.4}$$

In the following discussion, we specify the formula for calculating each component of Equation 4.

The prior $P(\tau_{max}|T)$ is 1 because we set the τ_{max} value when we find the potential parents for each variable.

We are using the same assumption in [227] that the networks change smoothly over time. We use the exponential priors on the change of network structures. We transform the form of the sequence of graph structures $G^T : G^T = (G_1, \ldots, G_m)$ into $G^T : G^T =$

$(G_1, \triangle G_1, \ldots, \triangle G_{m-1})$, where $\triangle G_i : 1 \leq i \leq m-1$ is the change of edges between G_i and G_{i+1}. we calculate $P(G^T|m,T)$ as follows.

$$\begin{aligned} P(G^T|m,T) &= P(G_1, \triangle G_1, \ldots, \triangle G_{m-1}) \\ &\propto P(G_1) \prod_{i=1}^{m-1} e^{-\lambda_s s_i} \\ &\propto P(G_1) e^{-\lambda_s \sum_{i=1}^{m-1} s_i} \\ &\propto P(G_1) e^{-\lambda_s S} \end{aligned} \tag{4.5}$$

,where $S : S = \sum_{i=1}^{m-1} s_i$, and s_i is the number of edges' change between G_{i+1} and G_i. We have no prior knowledge on $P(G_1)$ and see the uniform distribution as the prior.

We set the exponential prior on the transition times of networks over time and calculate $P(m|T)$ as the following.

$$P(m|T) \propto e^{-\lambda_m m} \tag{4.6}$$

We assume that the segments are independent and calculate $P(D_h|G^T, m, \tau^T, \tau_{max}, T)$ of each segment as the following.

$$P(D_h|G_h, \tau_h, \tau_{max}, T) = \int P(D_h|G_h, \tau_h, \tau_{max}, \Theta_{G_h}, T) \rho(\Theta_{G_h}|G_h) d\Theta_{G_h} \tag{4.7}$$

I_h is a segment where a network structure G_h and its corresponding lag value τ_h work. Θ_{G_h} are the parameters associated with the data of one segment I_h corresponding to G_h. $\rho(\Theta_{G_h}|G_h)$ is the probability density function of Θ_{G_h}.

We assume that the data are complete and multinomially distributed with a Dirichlet prior on the parameters. We weight the hyperparameters of Dirichlet distribution in each segment with the ratio of the segment length over the sample size. We calculate the BDe

[117] score of each segment as the following:

$$\begin{aligned} P(D_h|G_h,\tau_h,\tau_{max},T) &= \int P(D_h|G_h,\tau_h,\tau_{max},\Theta_{G_h},T)\rho(\Theta_{G_h}|G_h)d\Theta_{G_h} \\ &= \prod_{i=1}^{n}\prod_{j=1}^{q_{ih}}\frac{\Gamma(\alpha_{ij}(I_h))}{\Gamma(\alpha_{ij}(I_h)+N_{ij}(I_h))} \\ &\quad \prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}(I_h)+N_{ijk}(I_h))}{\Gamma(\alpha_{ijk}(I_h))} \end{aligned} \tag{4.8}$$

N is the sample size of the observed data. $|I_h|$ is the length of the segment I_h. Θ_{G_h} are the multinomial parameters of the joint probability distributions corresponding to G_h. r_i is the number of possible discrete values of x_i. q_{ih} is the number of configurations of parents π_i for the variable x_i in the segment I_h. $N_{ijk}(I_h)$ is the times that x_i had value k in the segment I_h. $N_{ij}(I_h) = \sum_{k=1}^{r_i} N_{ijk}(I_h)$. $\alpha_{ijk}(I_h)$ and $\alpha_{ij}(I_h)$ are the hyperparameters for Dirichlet distributions applied in the segment I_h. $\alpha_{ijk}(I_h)$ is assumed to be uniformly distributed inside a segment and is set to $\alpha_{ijk}(I_h) = \alpha|I_h|/(r_i q_{ih} N)$. α is the equivalent sample size.

We calculate the marginal likelihood $P(D|G^T, m, \tau^T, \tau_{max}, T)$ by using the modified Bayesian-Dirichlet equivalent (BDe) metric introduced in [227]. By multiplying the BDe metric of each segment, we get the extended BDe metric equation as follows:

$$\begin{aligned} P(D|G^T,m,\tau^T,\tau_{max},T) &= \prod_{h=1}^{m} P(D_h|G_h,m,\tau_h,\tau_{max},T) \\ &= \prod_{i=1}^{n}\prod_{h=1}^{m}\prod_{j=1}^{q_{ih}}\frac{\Gamma(\alpha_{ij}(I_h))}{\Gamma(\alpha_{ij}(I_h)+N_{ij}(I_h))} \\ &\quad \prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}(I_h)+N_{ijk}(I_h))}{\Gamma(\alpha_{ijk}(I_h))} \end{aligned} \tag{4.9}$$

Once the parents are decided, we use a conditional probability vector $\vec{p_\tau} = (p_1, \ldots, p_{\tau_{max}})$ with $\sum_{i=1}^{\tau_{max}} p_i = 1$. So $P(\tau^T|m, \tau_{max}, T)$ is calculated by:

$$P(\tau^T|m,\tau_{max},T) = \prod_{j=1}^{m} p_{\tau_j} \tag{4.10}$$

where p_{τ_j} is the conditional probability of the jth component's value in the lag vector τ^T.

4.3.1 Potential regulator detection

We know that the change of expression level of most transcriptional factors (TFs) always precedes or happens simultaneously with that of target genes[277]. This fact provides a useful technique to find potential regulators and relative expression lag value τ. We follow Zou's work [286] to detect the possible TFs.

In Zou's work, they used the expression levels of $\geq$ 1.2-fold and $\leq$ 0.70-fold compared with the average gene expression level as up-regulation and down-regulation cutoff thresholds. Any gene with initial up(down) change of expression level earlier is seen as the potential TFs of genes with change of expression level later. One example of up-regulation is showed in Figure 1. Instead of using a fixed value we relax the cutoff thresholds by taking a range of values. For up-regulation, we use the range $1.0 \sim 1.2$, and for down-regulation, we take the range $0.6 \sim 0.8$. In order to get all the possible TFs for each gene, we need to consider all the combinations of possible up(down)-regulation pairs. The yeast cell cycle data set analyzed by Zou has a limited time points ($T = 16$), which makes the complete search over all possible lag values affordable. However, with the increasing sample size and number of genes in the gene expression profiles, this searching algorithm is unrealistic and will bring more noises and high computational cost. We developed a heuristic to limit the potential regulator-target gene pairs for processing large data sets.

Below is our method. We first discretize the expression data by following the method above. We then search the data and only select the initial up(down)-regulation points. Slide the window with the width τ_{max} from the start($t = 1$) of the time series expression data to the end ($t = T - \tau_{max} + 1$), where T is the length of time points. For each moving step, the window slides one time step and only the up(down)-regulation pairs inside the window are calculated. One example of the sliding window is showed in Figure 2. We group the

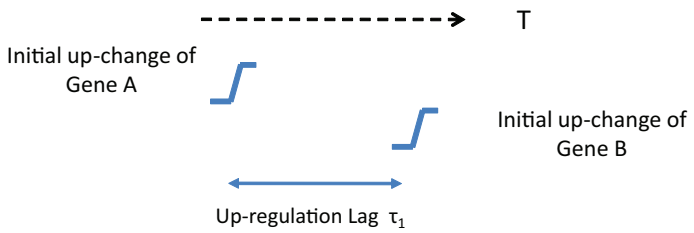


Figure 4.1: One example of detecting a potential up-regulation pair $A \rightarrow B$.

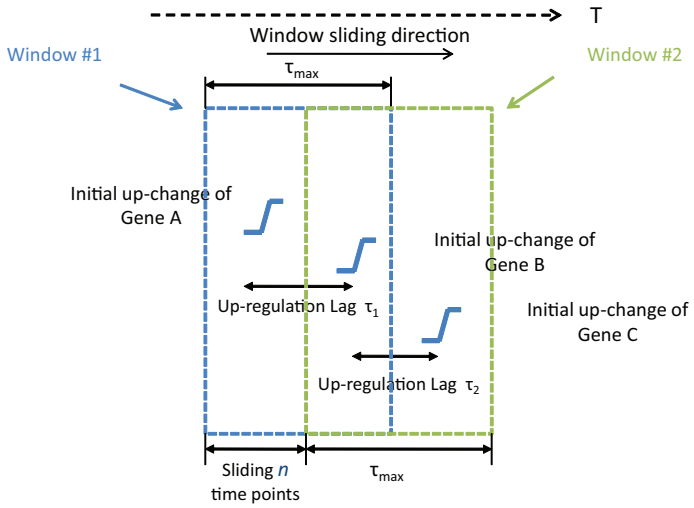


Figure 4.2: One example of the sliding window. With window 1, we found the potential up-regulation pair $A \rightarrow B$. After sliding n time points, with window 2, we identified $B \rightarrow C$.

pairs according to their time lag and calculate the posterior probability for each lag value $\tau : 1 \leq \tau \leq \tau_{max}$. For each gene, its potential TFs are also collected to be used as the prior knowledge to limit the search space during the process of structure sampling.

4.3.2 Structure sampling using RJMCMC

We choose sampling approaches rather than heuristic methods to search network structures due to the reason that microarray expression data are usually sparse, which makes the posterior probability of structures to be diffuse [129]. In this approach, a group of most likely structures could explain data better than a single one. We use a sampling method called RJMCMC (Reversible Jump Markov Chain Monte Carlo) to collect structure samples. The details of this method are available on [100].

Compared with the move types introduced in [227], we add one new move type called

change lag and modify most of the existing operations by incorporating more restrictions. We also define a vector of time points $L^T = (L_1, \ldots, L_{m-1})$, where $L_i : 1 \leq i \leq m-1$ is the start time point where G_{i+1} is applied. We use Metropolis-Hastings algorithm for RJMCMC sampling [50]. The move set of our RJMCMC consists of 11 move types:

MT1: *add edge to* G_i.

MT2: *delete edge from* G_i.

MT3: *add edge to* $\triangle G_i$.

MT4: *delete edge from* $\triangle G_i$.

MT5: *move edge between* $\triangle G_i$s.

MT6: *shift time*, which changes a single L_i's value. This operation will trigger the checking of τ_i's value under the restriction of $\tau_i \leq L_i - 2$, where $1 \leq i \leq m-1$, and $\tau_m \leq T - 1$.

MT7: *change lag*, which changes a single τ_i's value. This move type needs to follow the limitations showed on MT6.

MT8: *merge* $\triangle G_i$ *and* $\triangle G_{i+1}$.

MT9: *split* $\triangle G_i$.

MT10: *create new* $\triangle G_i$.

MT11: *delete* $\triangle G_i$.

Both MT8 and MT9 operations will trigger the change of dimensions of L^T and τ^T. In MT8, the new component of τ^T takes the least value of two merged components. Similarly with MT8 and MT9, M10 and M11 will change the dimensions of L^T and τ^T. MT1, MT3, MT10 and MT11 follow the restriction that the edges pointed to one target gene should have the origins from its potential regulators.

4.4 Experimental study and evaluation

We performed all the experiments on a cluster with 256 Intel Xeon 3.2 Ghz EM64T processors with 4 GB memory each. We implemented our method FLnsDBNs (Flexible Lag Non-Stationary Dynamic Bayesian Networks) in Matlab.

We compare three approaches: our approach FLnsDBNs, reversible jump Markov chain Monte Carlo Non-Stationary Dynamic Bayesian Networks (RJnsDBNs) [227], and Allocation Sampler Non-Stationary Dynamic Bayesian Networks (ASnsDBNs) [102]. For RJnsDBNs, we use the default setting of unknown numbers and times of transitions (UNUT) in all of the data sets. RJnsDBNs is implemented in Java, and ASnsDBNs is implemented in Matlab. We show the average elapsed time of three methods on two data sets in Table 1. In FLnsDBNs, we ignore the computational cost on the potential regulator detection process because it takes less than 0.03 second. Although the direct comparison of three approaches by using the elapsed time is unfair due to the difference in implementation, our method shows the comparable computational performance with ASnsDBNs.

Our experimental study is based on three data sets: (i) Bone Marrow-derived Macrophages gene expression time series data (Macrophages data set), (ii) Circadian regulation in Arabidopsis Thaliana gene expression time series data (Arabidopsis data set), and (iii) Drosophila muscle development gene expression time series data (Drosophila data set). To compare the results from different data sets, we follow the evaluation method introduced in [129, 262, 102]. For each data set, we first collect gold standard reference networks as the ground truth. For the Macrophages data set, such reference networks are available in [148, 224, 102]. For the Arabidopsis data set, we collect the network information from [183, 228, 62, 110, 190]. For the Drosophila data set, there is no ground truth regarding the network structure. We compare our method with others by showing the commonality and differences. In case where we have ground truth network structure (the Bone Marrow data set and Arabidopsis data set), we use the area under receiver operating characteristic curve (AUROC) values to evaluate the performance. We obtained the ROC curves by postprocessing the posterior probabilities

of directed edges and taking different cutoff thresholds in $[0, 1]$. If the posterior probability of an edge is greater than the threshold, we keep the edge. Otherwise, we do not keep the edge. With the ROC curves, we evaluate the performance of different methods by comparing the AUROC scores. In addition, for each data set, we show the posterior distribution of the number of segments and the locations of changepoints. In all of our experimental study, we find that the method FLnsDBNs produces compatible results with previous methods and demonstrates better network prediction performance in all the data sets. Before we discuss the details of experimental results, we present our data set first below.

4.4.1 Data sets

As mentioned briefly before, we evaluate our method on three data sets used in [227, 102]. We preprocess the original data sets by following Zhao's work [283]. We set the values of a missed time point with the mean of its two neighbors; i.e., $X_{i,t} = (X_{i,t-1} + X_{i,t+1})/2$ if $1 < t < T$. If the missed values are at the beginning or end, simply set the same value as its neighbor; i.e., $X_{i,t} = X_{i,t+1}$ if $t = 1$ or $X_{i,t} = X_{i,t-1}$ if $t = T$. In the following, we show the details of each data set.

Bone Marrow-derived Macrophages gene expression data. Interferon regulatory factors (IRFs) are proteins crucial for the mammalian innate immunity [120]. These transcription factors are central to the innate immune response to the infection by pathogenic organisms [224]. We use the Macrophage data sets sampled from three external conditions: (I) Infection with Cytomegalovirus (CMV), (II) Treatment with Interferon Gamma (IFN_γ), and (III) Infection with Cytomegalovirus after pretreatment with IFN_γ ($CMV{+}IFN_\gamma$). Each data set has 3 genes: *Irf1*, *Irf2* and *Irf3*, and contains 25 time points with the interval of 30 minutes. We use the network $Irf2 \leftrightarrow Irf1 \leftarrow Irf3$ as the gold standard and assume the network never changes over the time.

Arabidopsis thaliana circadian regulation gene expression data. A. thaliana circadian gene expression data was sampled to understand the internal clock-signalling network

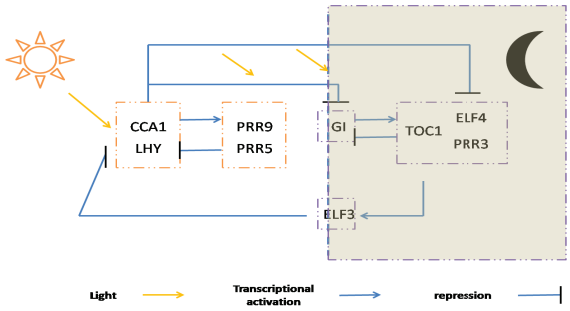


Figure 4.3: The A. thaliana oscillator loops of the circadian clock network.

of plant. Two data sets were collected with the interval of 2h from two light-dark conditions: 10h:10h and 14h:14h light/dark cycles, both of which contain 13 time points. We choose a group of 9 genes, *LHY*, *CCA1*, *TOC1*, *ELF4*, *ELF3*, *GI*, *PRR9*, *PRR5*, and *PRR3* for analysis, which create transcriptional feedback loops. We show the referred biological regulatory network in Figure 3. In this network, *CCA1*, *LHY* and *TOC1*, as core components of the reciprocal regulation , are important for the proper function of this oscillator network in A. thaliana [183]. *CCA1* and *LHY* proteins' direct binding to the promoter of *TOC1* represses the expression of *TOC1*, and *ELF3* works as a negative regulator of light signaling to the clock oscillator and enables the induction of oscillator output [228, 62]. The pseudo-response regulators PRR5 and PRR9 are activated by *CCA1* and *LHY* accompanied with light, and repress *CCA1* and *LHY* subsequently. *G1* is activated by light and improve the expression of *TOC1*. *ELF4* is repressed by *CCA1*. And *PRR3* is highly correlated with *TOC1* and together form a functional complex [208].

Drosophila muscle development gene expression data. The original transcriptional profile on the life cycle of Drosophila melanogaster contains 4028 genes, nearly one third of all of the predicted Drosophila genes. The samples were collected over 66 time steps throughout the life cycle of Drosophila melanogaster consisting of four periods: embryonic, larval, pupal, and adulthood periods [16]. The intervals of sampling are not even, from overlapped 1 hour during the early embryonic period to multiple days in the adulthood. We choose 11 genes for analysis, which are *eve*, *gfl/lmd*, *twi*, *mlc1*, *sls*, *mhc*, *prm*, *actn*, *up*, *myo61f*, *msp300*. Those genes were reported to be related with the muscle development of

Drosophila.

4.4.2 Experimental results

In this section, we compare the experimental results of three approaches: FLnsDBNs, RJnsDBNs, and ASnsDBNs on three data sets.

The experimental results on Macrophages data. On the Macrophages data, for each method, we run 10,000 iterations for burn-in and then take additional 40,000 iterations to collect samples. In Figure 4, 5 and 6, we show the posterior probabilities of the numbers of segments and changepoints on three Macrophages data sets. The sample collection of FLnsDBNs on the Macrophages data takes about 2 minutes.

For the CMV data, we first observe that there is a high agreement among all three methods in term of the range of the number of identified segments. The ranges are $1 \sim 4$ for FLnsDBNs, $1 \sim 4$ for RJnsDBNs, and $2 \sim 4$ for ASnsDBNs. When we compare the distributions of the number of segments identified by three methods, we observe that ASnsDBNs clearly identifies a dominant 3-segment in the data set while the posterior probabilities produced by FLnsDBNs and RJnsDBNs are flat. For the predicted locations of the changepoints, FLnsDBNs identifies three posterior peaks at time stamps 4, 8, and 14. RJnsDBNs finds four peaks at 5, 11, 14, and 19. In ASnsDBNs, two peaks happen at 1 and 4 with the probabilities more than 0.5. There is a consensus among three methods that the most probable changepoint occurs at the location 4. The results of three methods are consistent with the biological phenomenon that the simultaneous responses of Macrophages happen under the attack of Cytomegalovirus [102]. In order to assess the network prediction performance, we show the AUROC scores in Table 2. We find that all methods perform well in the CMV data with the AUROC scores equal to 1.

For the $CMV + IFN_\gamma$ data, all three methods identify 1 segment, which corresponds to a coexistence state between virus and its host cell [22, 102], and have the same range of the number of segments $1 \sim 3$. In Table 2, we find that FLnsDBNs shows a much better

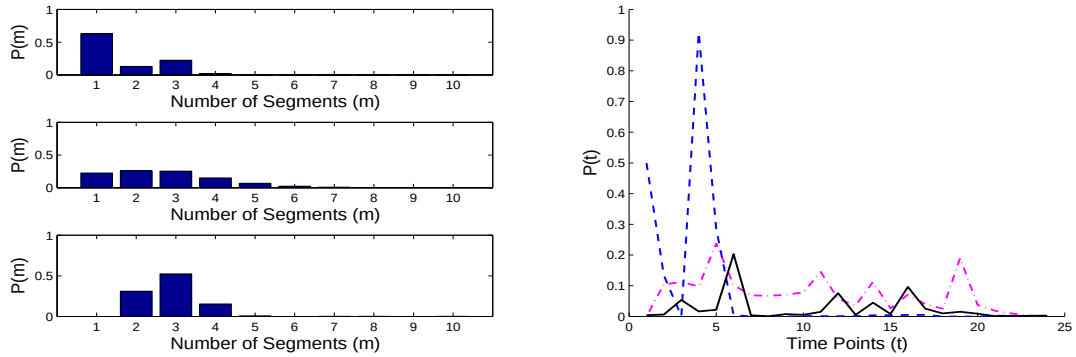


Figure 4.4: Comparison of three methods on CMV Macrophage data. Left: The posterior probabilities of the numbers of segments (top: FLnsDBNs ($\lambda_m = 4.05, \lambda_s = 2$); middle: RJnsDBNs ($\lambda_m = 0.65, \lambda_s = 2$); bottom: ASnsDBNs). Right: The posterior probabilities of the change points (FLnsDBNs: black solid line; RnsDBNs: magenta dash-dot line; ASnsDBNs: blue dashed line).

network prediction with the AUROC score equal to 1 while in RJnsDBNs the AUROC score is equal to 0.2222 and in ASnsDBNs the AUROc score is equal to 0.6667.

For the IFN_γ data, there is a postulated transition with the immune activation under the treatment of $IFN\gamma$. FLnsDBNs infers 2 segments and finds two posterior peaks of transition time at 8 and 14. ASnsDBNs and RJnsDBNs infer only one segment, even though the two methods identify a differnt posterior peak at the location around 5. On the assessment of the predicted network structures, the AUROC scores are 0.8333 in FLnsDBNs, 0.7778 in RJnsDBNs, and 0.6667 in ASnsDBNs. In all of three Macrophages data sets, our approach shows the best network prediction accuracy.

For each Macrophages data set using FLnsDBNs and RJnsDBNs methods, we find that the posterior probability distributions of any edge do not change much across different segments. This finding is consistent with the assumption that the underlying network does not change through the time.

The experimental results on Arabidopsis data. On the Arabidopsis data, we use a larger number of iterations in the MCMC sampling because the data set is much larger than the Macrophages data. We run 10,000 iterations for burn-in and then take additional 990,000 iterations to collect samples. The sample collection of FLnsDBNs on the Arabidopsis data takes about 4 hours.

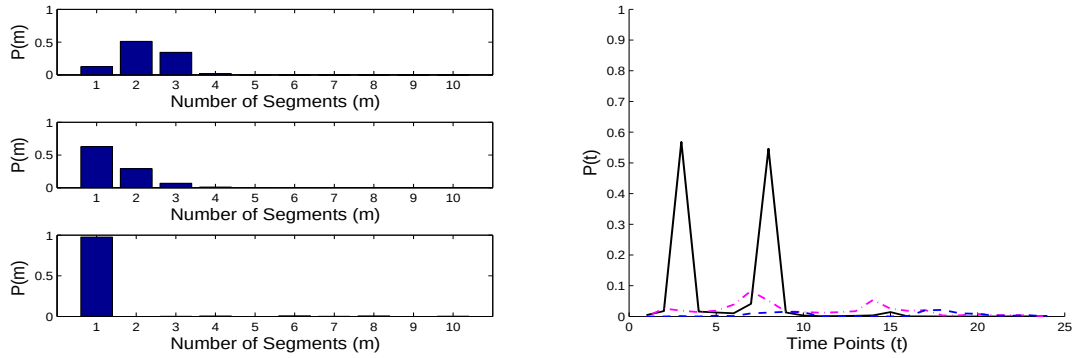


Figure 4.5: Comparison of three methods on $CMV+IFN_\gamma$ Macrophage data. Left: The posterior probabilities of the numbers of segments (top: FLnsDBNs ($\lambda_m = 6, \lambda_s = 2$); middle: RJnsDBNs ($\lambda_m = 1, \lambda_s = 2$); bottom: ASnsDBNs). Right: The posterior probabilities of the change points (FLnsDBNs: black solid line; RnsDBNs: magenta dash-dot line; ASnsDBNs: blue dashed line).

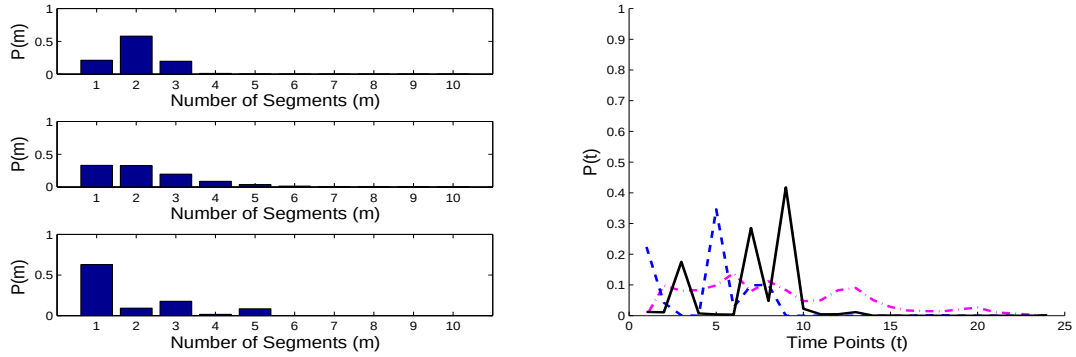


Figure 4.6: Comparison of three methods on IFN_γ Macrophage data. Left: The posterior probabilities of the numbers of segments (top: FLnsDBNs ($\lambda_m = 6.5, \lambda_s = 2$); middle: RJnsDBNs ($\lambda_m = 0.001, \lambda_s = 2$); bottom: ASnsDBNs). Right: The posterior probabilities of the change points (FLnsDBNs: black solid line; RnsDBNs: magenta dash-dot line; ASnsDBNs: blue dashed line).

In Figure 7 and 8, we show the posterior distributions of the numbers of segments and changepoints on two Arabidopsis data sets. For the Arabidopsis T20 data, in FLnsDBNs the range of the number of segments is $2 \sim 3$, and in RJnsDBNs and ALnsDBNs the ranges are $1 \sim 4$. In FLnsDBNs, the dominant samples are the ones with 2 segments while in AlnsDBNs they are 3 segments. For the Arabidopsis T28 data, the ranges are $2 \sim 3$ in FLnsDBNs, $1 \sim 3$ in RJnsDBNs and $3 \sim 5$ in ASnsDBNs. FLnsDBNs infers 2 segments, RJnsDBNs infers 1 segment, and ASnsDBNs infers 5 segments, respectively on the T28 data. In both data sets, we find that the differences of the posterior probabilities of 2 and 3 segments are low in RJnsDBNs and the difference between the posterior peaks of changepoints and the time points nearby are not noticeable. Hence, for this data set, we only use a single network in RJnsDBNs to compare with other methods. Using ASnsDBNs, the poseterior peaks of changepoints on T20 data are 1, 5 and those on T28 are 2, 7, 10. In [102], the results of ASnsDBNs are explained as a phase shift incurred by different dark/light cycles. However, our approach predicts the posterior peak of changepoints both at the location 6.

We evaluated the network reconstruction accuracy of three methods by comparing with the reference network showed in Section 3.2. We show the AUROC scores in Table 3. In addition, we use a new comparative criteria called the TP|FP=5 counts [262, 102] to further demonstrate the performance of our method. TP are the true positive counts; FP are the false positive counts; TP|FP=5 are the TP counts when FP is 5. The TP|FP=5 counts of three approaches are shown in Table 4. FLnsDBNs outperforms other two methods in both two evaluation criteria of the AUROC score and TP|FP=5 counts on the Arabidopsis data sets.

The experimental results on Drosophila data. For the Drosophila data, We run 10,000 iterations for burn-in and then take additional 990,000 iterations to collect samples. The sample collection of FLnsDBNs on the Drosophila data takes about 10 hours.

We show the results of posterior probabilities of the numbers of segments and changepoints in Figure 9. ASnsDBNs predicts more than 20 segments and fails to provide a mean-

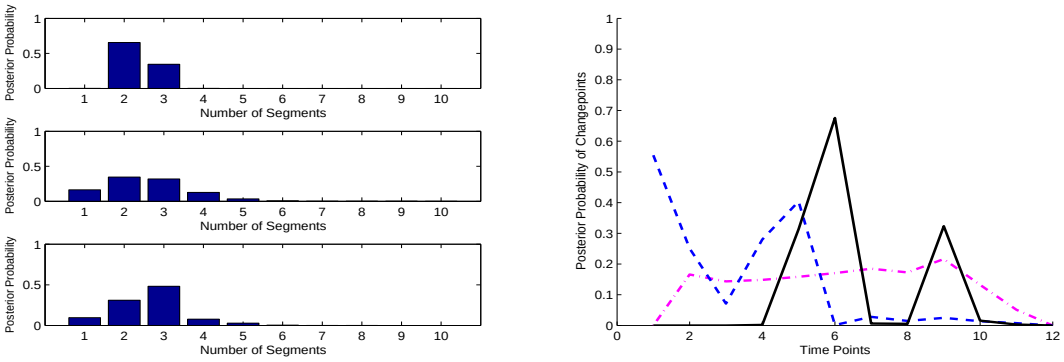


Figure 4.7: Comparison of three methods on Arabidopsis T20 data. Left: The posterior probabilities of the numbers of segments (top: FLnsDBNs ($\lambda_m = 14, \lambda_s = 2$); middle: RJnsDBNs ($\lambda_m = 0.0005, \lambda_s = 2$); bottom: ASnsDBNs). Right: The posterior probabilities of the change points (FLnsDBNs: black solid line; RnsDBNs: magenta dash-dot line; ASnsDBNs: blue dashed line).

Table 4.1: Comparison of AUROC values on Arabidopsis data

	*ArabidopsisT*20	*ArabidopsisT*28
RJnsDBNs	0.5070	0.5773
ASnsDBNs	0.5929	0.5641
FLnsDBNs	G1:0.6138; G2:0.6150	G1:0.6558; G2:0.6628

TP, true positive; FP, false positive; TN, true negative; FN, false negative.
Sensitivity = TP/(TP+FN).
Specificity = TN/(TN+FP).
Complementary Specificity = 1- Specificity = FP/(TN+FP).
The ROC curves are plotted with the Sensitivity scores against the corresponding Complementary Specificity scores. G1 and G2 are two networks reconstructed based on the changepoint 6.

Table 4.2: Comparison of $TP|FP = 5$ values on Arabidopsis data

	*ArabidopsisT*20	*ArabidopsisT*28
RJnsDBNs	2	6
ASnsDBNs	4	3
FLnsDBNs	G1:8; G2:8	G1:11; G2:11

G1 and G2 are two reconstructed networks separated by the changepoint 6.

ingful result of changepoints. Therefore, in the subsequent discussion, we only compare FLnsDBNs and RJnsDBNs approaches. The assumed transition time of four life periods are located at 30, 40 and 58. RJnsDBNs predicts 3 segments with the posterior peaks located at 11 and 21. FLnsDBNs prefers 4 segments with the posterior peaks at 19, 36 and 54, which happen before the assumed changepoints. And our prediction of the Embryonic→Larval

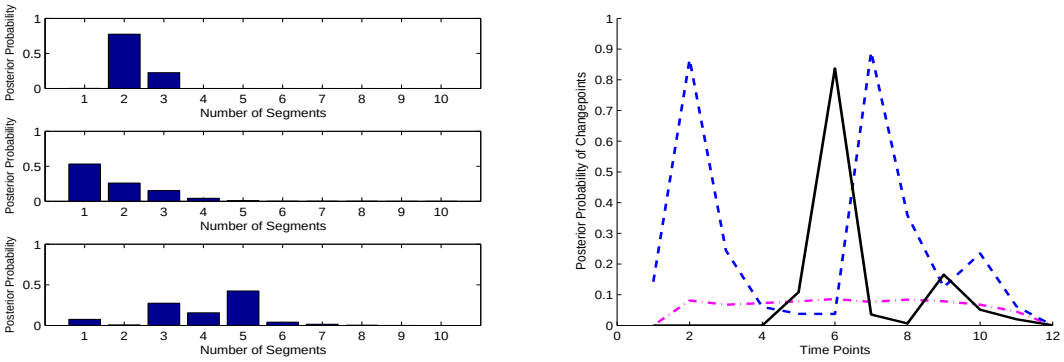


Figure 4.8: Comparison of three methods on Arabidopsis T28 data. Left: The posterior probabilities of the numbers of segments (top: FLnsDBNs ($\lambda_m = 14, \lambda_s = 2$); middle: RJnsDBNs ($\lambda_m = 0.005, \lambda_s = 2$); bottom: ASnsDBNs). Right: The posterior probabilities of the change points (FLnsDBNs: black solid line; RnsDBNs: magenta dash-dot line; ASnsDBNs: blue dashed line).

transition occurs at 19 much earlier than 30. Both ASnsDBNs and RJnsDBNs methods do not converge well in this fly data set.

We show the reconstructed networks of our approach, those of RJnsDBNs (UNUT), a stationary directed network predicted by [284], and the non-stationary undirected networks predicted by [105] in Figure 10 for the purpose of comparison. In addition, we provide the networks predicted by RJnsDBNs with another setting of KNKT to compare because the networks inferred by RJnsDBNs (UNUT) show much difference from other predictions. In the following, we only compare the results of [105], [284], RJnsDBNs (KNKT) and FLnsDBNs.

These four predictions share many similarities and also show some difference. We find that the gene *msp-300* may play a key role in the cluster of these 11 genes. *myo-61f* is only predicted to be a regulated gene by *msp-300* in [284], but other three methods show that *myo-61f* is another key gene in this cluster. In [284], *myo-61f* is correlated with *twi*, *sls*, *mlc1*, *mhc* and *msp-300*. In RJnsDBNs (KNKT), *myo-61f* serves as the regulators of *prm*, *up* and *sls*. Our approach predicts that *myo-61f* regulates four genes: *sls*, *prm*, *actn*, and *msp-300*. FLnsDBNs, [284] and [105] all agree that there are regulation relationships between *myo-61f* and *msp300*, while RJnsDBNs (KNKT) did not identify this interaction. Different from the prediction of RJnsDBNs (KNKT), Our approach finds that *twi* is not

separated from other genes and *actn* serves as the parents of other genes, which is consistent with the networks in [284]. In Figure 10E, *twi* is the regulator of *sls*, and *actn* regulates *sls*, *prm* and *gfl*. We also notice that the regulating effects of *myo-61f* and *msp-300* on other genes intensify over the time. Nearly different from all of three methods, our approach finds that *twi* and *gfl/lmd* are regulators of other genes while only [284] sees *twi* as a regulator. *gfl/lmd* and *twi* are direct upstream regulators of *mef2* [74, 63] that directly regulates some target myosin family genes at all stages of muscle development [229] , such as *mhc* and *mlc1*. Evidence show the cooperative binding of *twi* and *Mef2* or *gfl/lmd* and *Mef2* to these target genes are attractive models [229, 74]. It indicates that a co-regulation role of *twi* and *gfl/lmd* with *Mef2* to other muscle development genes may exist. The prediction of our method shows this biological behavior. Currently the reference regulatory network on the muscle development of Drosophila melanogaster is not available and the relevant biological literatures are limited. Further biological researches and experiments are needed to verify the regulatory networks.

Chapter 5

Preliminary Work (III): Non-Stationary Dynamic Bayesian Networks based on Perfect Simulation

We proposed a novel non-stationary DBNs method [145]. Our method is based on the perfect simulation model. We applied this approach for the gene regulatory network inference on three non-stationary time series data and compared with other two non-stationary DBNs methods. The experimental results demonstrated that our method outperformed two other state-of-the-art methods in both computational cost and structure prediction accuracy. The further sensitivity analysis showed that once converged our model is insensitive to the parameter, which reduces the uncertainty of the model behavior.

5.1 Introduction

Non-stationary Dynamic Bayesian Network methods are widely used to model the temporal changes of dependency structures from multivariate time series data [227, 102, 240, 132, 170, 70, 130, 103]. Comparing to traditional DBNs modeling, non-stationary DBNs have advantages to capture the structural dynamics of networks in various biological systems, such

as Neural assemblies in response to visual stimuli [240], morphogenesis in the organisms' life cycle [227, 130], adaptive mammalian immune response against infection of virus [102], or circadian regulation dynamics of plants caused by dramatic changes of outside environment such as light intensity [102].

Several methods have been developed for constructing non-statio- nary models. For example, Robinson et al. proposed a discrete non-stationary DBNs method [227] using Reversible Jump Markov Chain Monte Carlo (RJMCMC) [100] to sample underlying changing network structures. Grzegorczy et al. proposed a non-homogeneous continuous Bayesian network method with a Gaussian mixture mod- el based on the allocation sampler technique [204]. Grzegorczy et al. improved the convergence of their method using perfect simulation modeling [77] and reduced the risk of overfitting and inflated inference uncertainty [130] in their later work [103]. Both Robinson's and Grzegorczyk's methods perform change-point detection and we call them change-point based approaches. Song et al. [240] proposed a time-varying DBNs (TV-DBNs) method and used a kernel re-weighted l_1-regularised auto-regressive approach for learning the graph structures at each time step. Lebre et al. [170] proposed a more flexible auto-regressive time varying model called ARTIVA that allows gene-by-gene analysis. Husmeier et al. [130] introduced inter-time segment information sharing schemes to address the over-flexibility issue in the ARTIVA approach. Those three approaches are different from the change-point detection based approaches and fell into the category of structure learning of constantly varying network over time. In this paper, our work focuses on the change-point detection modeling for regulatory network dynamics.

There are several limitations of the existing change-points based techniques. First, the mixture model used by Grzegorczy et al. [102, 103] assumed that the underlying network structures are invariant over time. Such an assumption is too rigid when changes of network structures are expected, for example, morphogenesis or embryogenesis [130]. Second, Grzegorczy's method with the improvement on convergence [103] mixed the structure sampling steps and the perfect simulation steps in the same RJMCMC procedure. The time complex-

ity of each perfect simultion step is quadratic to the number of the observations [77]. This scheme brings extra computational costs on change-point simulation that are proportional to the number of sampling iterations even if genes are decomposed into groups to alleviate the computational concern. Third, the RJMCMC sampling approach in Robinson's work [227] converges slowly. For example, the results in [100] using RJMCMC did not converge as pointed out in the subsequent work in [101]. In addition, our experiments show that the structure prediction accuracy of Robinson's RJMCMC is low.

We posit that the key computational obstacle for efficient modeling of time series data with non-stationary DBNs methods is the interplay of change-point detection and structure inference for each identified time segment. To improve computational efficiency, we designed an algorithm called ReCursion Non-Stationary Dynamic Bayesian Networks (RCnsDBNs) to separate these two essential steps. Our method adopted Fearnhead's perfect simulation model [77] for change-point detection. The perfect simulation model was originally developed for univariate time series data [77] and we modified the algorithm to model our multi-variate time series data. In particular, we designed an iterative algorithm for the structure inference and change-point detection. Our method first used the point process [215] as the prior for the occurrences of change-points and directly simulated the change-points from the posterior distribution. For each predicted segment, we then used a regular Markov Chain Monte Carlo (MCMC) method, a revised KNUT (Known Transition Number Unknown Transition Time) setting in Robinson's method [227]. Once the algorithm converges, we output the most likely change-points and a sequence of network structures corresponding to the separated segments.

There are several advantages for the novel non-stationary DBNs algorithm. First, by directly simulating the posterior distribution of transition time for graph structures, our method efficiently reduces the model space and improves the computational performance both on time and numbers of sampling iterations for covergence. Second, even if a negative binomial prior is adopted in our point process, the experiments showed that our experimen-

tal results are stable within large parameter ranges, which reduces the uncertainty of the model behavior. Third, different from Grzegorczy's method [103], our method only needs to simulate the change-points once for each round of our algorithm. It saves the computational time. Fourth, Our approach outperform Robinson's RJMCMC approach on structural prediction accuracy. Even if our discrete model needs the discretization of the data, compared with Grzegorczyk's continuous approach, our method showed the competitive performance for structure estimation.

5.2 Related Work

The change-point detection problems have been extensively investigated in time series models. Recent work could be found in: PCA-based singular-spectrum transformation models [192]; non-parametric online-style algorithm via direct density-ratio estimation [153]; two-phase linear regression model [178]; a hybrid algorithm that relies on particle filtering and Markov chain Monte Carlo [54]; the RJMCMC method [100]; The perfect simulation model based on product-partition model [77]; change-point detection by minimizing a penalized contrast function [169]. These models are widely used in various applications, such as climate analysis [178], coal-mining disaster analysis [100, 77], well-log analysis [77], the analysis of abrupt economic agents' behaviors [54], and asset price volatility [169].

Researchers find that change-point modeling is a very promising way of dealing with the non-stationarity property [54]. Hence, the current non-stationary DBNs methods employed different change-point detection techniques to model the underlying change-point processes of network structures. Robinson et al. [227] applied RJMCMC [100] and used a discrete model with the assumed multinomial distributed data with the Dirichlet prior. Using the RJMCMC technique, Lebre et al. [170] proposed a new time varying networks approach based on first-order auto-regression and Yao's two-stage regime-SSM model [222]. Their method focuses on local structural changes and performs node-by-node analysis. Fur-

ther, in order to address the structural overfitting problem in [170], Dondelinger et al. [70] and Husmeier et al. [130] introduces information sharing between segments into Lebre's approach. Grzegorczy et al. [102, 103] applied the allocation sampler technique and introduced a continuous-valued DBNs method that approximates the non-stationary property with a Gaussian mixture model. Based on their work, Ickstadt et al. [132] further generalized this non-linear BGe mixture model into a broader framework of non-parametric Gaussian Bayesian networks. In this paper, we incorporate the perfect simulation modeling into our dynamic bayesian framework and provide a computationally efficient non-stationary DBNs approach. We chose this change-point detection technique for the following reasons. First, perfect simulation is based on bayesian analysis and can be easily applied into our MCMC algorithm. Second, with an approximation in the recursion, the computational complexity of this method is approximately linear to the number of observations.

5.3 Methods

5.3.1 Perfect Simulation Modeling

Fearnhead used the perfect simulation model to find change-points in the univariate time series data [77]. We adapted his method to the framework of our dynamic bayesian networks, and provided a non-stationary DBNs method to detect the change-points for network structures in multivariate time series data.

We consider an observed time series data $D = \{y_1, \cdots, y_T\}$ spanning T time points, where each observation $y_i \in \mathbf{R}^n : 1 \leq i \leq T$ is a n dimensional vector $(x_1, \cdots, x_n)$. The time series data is subdivided to m segments $D = \{D_1, \cdots, D_m\}$, where m is unknown. We denote the change-points for these segments as $L^T = (l_0, l_1, \ldots, l_{m-1}, l_m)$, where $l_0 = 0$ and $l_m = T$.

We assume the change-points as a point process on positive integers, which is characterized by a probability mass function $g(t)$, where t is the distance of two successive change-

points. We choose the negative binomial distribution as the distribution for the distance between two successive change-points and have $g(t) = \binom{t-1}{k-1} p^k(1-p)^{t-k}$ with the parameters $k > 0, p > 0$ and its corresponding accumulative distribution function $G(t) = \sum_{i=1}^{t} g(i)$. For the special case of the first change-point, we have $g_0(t) = \sum_{i=1}^{k} \binom{t-1}{i-1} p^i(1-p)^{t-i}$ and $G_0(t) = \sum_{i=1}^{t} g_0(i)$.

Given the assumption of the independence between segments, we calculate the probability of a sequence of observations after one change-point l: $Q(t) = Pr(y_{t:n}|l = t-1)$ by using recursive function below:

$$Q(t)=\sum_{s=1}^{T-1} P(t,s)Q(s+1)g(s+1-t)+P(t,T)(1-G(T-t)). \tag{5.1}$$

where $2 \leq t \leq T$, and

$$Q(1)=\sum_{s=1}^{T-1} P(1,s)Q(s+1)g_0(s)+P(1,T)(1-G_0(T-1)). \tag{5.2}$$

In Equation 1 and 2, $P(t,s)$ is the simplified notation of $P(y_{t:s}|l = t-1) : 1 \leq t \leq T, t \leq s \leq T$, where the observations $y_{t:s}$ are in the same segment between two change-points $t-1$ and s. Similarly, $P(1,s)$ is the simplified notation of $P(y_{1:s}|l = 0) : 1 \leq s \leq T$, where the observations $y_{1:s}$ are in the same segment between two change-points l_0 and s.

Further, based on $Q(t)$, we calculate the probability distribution of the first change-point below:

$$P(l_1)=P(1,l_1)Q(l_1+1)g_0(l_1)/Q(1). \tag{5.3}$$

where $l_1 : 1 \leq l_1 \leq T-1$ is the first change point.

Given l_i, we calculate the conditional probability $P(l_{i+1}|l_i) : l_i + 1 \leq l_{i+1} \leq T-1$ as the

following:

$$P(l_{i+1}|l_i)=P(l_i+1,l_{i+1})Q(l_i+1)g(l_{i+1}-l_i)/Q(l_i+1). \tag{5.4}$$

And the probability of no more change-point is given as:

$$P(T|l_i)=P(l_i+1,T)(1-G_0(T-l_i-1))/Q(l_i+1). \tag{5.5}$$

Finally, with the probability distribution of l_1 and conditional distribution of l_{i+1} given l_i, we directly simulate the change-point samples and compute the posterior probability distributions $P(L^T|T)$ and $P(m|T)$. Due to the limitation of the space, we omitted the mathematical derivation and proofs. More technical details are available on [77]. One of the key computations in the simulation procedure is to compute the probability $P(y_{t1:t2}|l = t1-1) : 1 \leq t1 \leq T, t1 \leq t2 \leq T$. By assuming the i.i.d. observations in a single segment and the conjugate priors $\rho(\theta)$ on the parameters θ associated with each segment, Fearnhead's work provides a closed form of solution for $P(y_{t1:t2}|l = t1-1) = \int \prod_{i=t1}^{t2} f(y_i|\theta)\rho(\theta)d\theta$. In the analysis of the well-log data, he assumed the normally distributed observations: $y_i \sim N(\mu_i, \sigma^2)$ with the fixed variance σ^2 and a normal prior for the mean μ_i. In the following we discuss our solutions both under the static and dynamic bayesian network frameworks.

Perfect simulation modeling in bayesian networks. Bayesian networks (BNs) are a special case of probabilistic graphic models. A static BN is defined by an acyclic directed graph G and a complete joint probability distribution of its nodes $P(X) = P(X_1, \ldots X_n)$. The graph $G : G = \{X, E\}$ contains a set of variables $X = \{X_1, \ldots, X_n\}$, and a set of directed edges E, defining the causal relations between variables. With a directed acyclic graph, the joint distribution of random variables $X = \{X_1, \ldots, X_n\}$ are decomposed as $P(X_1, \ldots, X_n) = \prod_i P(X_i|\pi_i)$, where π_i are the parents of the node (variable) X_i.

We assume that the observations inside one segment are independent. In each segment there is one graph $G_h : 1 \leq h \leq m$ that dominates the segment. We denote $y_{t1:t2}$ as $D^*_{t1:t2}$

and calculate $P(D^*_{t1:t2}|l = t1-1)$ as:

$$P(D^*_{t1:t2}|l{=}t1{-}1){=}\textstyle\sum_G P(y_{t1:t2}|l{=}t1{-}1,G)P(G) \tag{5.6}$$

$$={\textstyle\sum_G} P(G)\prod_{j=t1}^{t2} P(y_j|G){=}\textstyle\sum_G P(G)\prod_{j=t1}^{t2}\prod_i P(x_i^j|\pi_{x_i})$$

$$={\textstyle\sum_G} P(G)\int P(D_{t1:t2}|G,\Theta_G)\rho(\Theta_G|G)d\Theta_G$$

Θ_G are the parameters associated with the data $D_{t1:t2}$ corresponding to G. $\rho(\Theta_G|G)$ is the probability density function of Θ_G.

Under the assumption that the data are complete and multinomially distributed with a Dirichlet prior on the parameters Θ_G, we have the BDeu [117] solution to $P(y_{t1:t2}|l = t1-1)$:

$$P(D^*_{t1:t2}|l{=}t1{-}1,G){=}\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \tag{5.7}$$

$$\prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})}$$

r_i is the number of possible discrete values of x_i. q_i is the number of configurations of parents π_i for the variable x_i. N_{ijk} is the times that x_i had value k. $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. α_{ijk} and α_{ij} are the hyperparameters for Dirichlet distribution. α_{ijk} is assumed to be uniformly distributed inside a segment and is set to $\alpha_{ijk} = \alpha/(r_i q_i)$. The equivalent sample size α is set to 1.

In order to calculate $P(D^*_{t1:t2}|l = t1-1)$, we need to provide a model space $\mathcal{M}$ for G. We use MCMC to simulate $\mathcal{M}$. Given the collected sample size $N_{\mathcal{M}}$, we approximate the calculation of Equation 6 as follows:

$$P(D^*_{t1:t2}|l{=}t1{-}1){=}\textstyle\sum_{i=1}^{N_{\mathcal{M}}}\frac{1}{N_{\mathcal{M}}}P(D^*_{t1:t2}|l{=}t1{-}1,G) \tag{5.8}$$

Perfect simulation modeling in dynamic bayesian networks. The topology of bayesian networks must be a directed acyclic graph and hence could not be used to model the case where two nodes may be dependent on each other. As an extension of BNs to model time series data, Dynamic Bayesian Networks (DBNs) lift the limitation of directed

acyclic graph by incorporating temporal dependence in constructing bayesian networks. It is not straightforward to extend the solution in BNs to modeling DBNs mainly due to that neighboring observations are not independent given the model parameters. Below we develop a heuristic and provide a solution for $P(D^*_{t1:t2}|l = t1-1)$ as follows.

We set the lag value $\tau = 1$ and assume the segments are overlapped. For each segment $D_h : 1 < h \leq m$, the length of overlapped area with the previous segments $D_1, \cdots, D_{h-1}$ is equal to the lag value $\tau = 1$. For D_1, there are no previous segments and we add $\tau = 1$ additional y_1s at the beginning of D_1. Given the transition time points L^T, each segment of observations $D_h = \{y_{l_{h-1}-\tau+1}, \cdots, y_{l_h}\}$, where $y_i = y_1$ when $i \leq 0$. We denote $\{y_{l_{h-1}+1}, \cdots, y_{l_h}\}$ as D^*_h and $\{y_{l_{h-1}-\tau+1}, \cdots, y_{l_{h-1}}\}$ as D^{*c}_h, and have $D_h = \{D^*_h, D^{*c}_h\}$. We take a heuristic to assume that D^{*c}_h is independent of G_h, and that $P(D^{*c}_h)$ is always equal to 1. Similarly we denote $y_{t1:t2}$ as $D^*_{t1:t2}$, $y_{t1-\tau:t1-1}$ as $D^{*c}_{t1:t2}$, and $D_{t1:t2} = \{D^*_{t1:t2}, D^{*c}_{t1:t2}\}$. In each segment, there is one graph $G_h : 1 \leq h \leq m$ that dominates the segment.

With the assumption that

$$P(D^c_h)=\begin{cases} 1 & \text{if } D^c_h = D^{*c}_h \\ 0 & \text{otherwise} \end{cases}$$

, we have Theorem 1

Theorem 3.

$$P(D^*_h)=P(D^*_h|D^{*c}_h)=P(D_h) \tag{5.9}$$

Proof.

$$P(D^*_h)=\sum_{D^c_h} P(D^*_h|D^c_h)P(D^c_h)$$

$$=P(D^*_h,D^{*c}_h)=P(D_h)=P(D^*_h|D^{*c}_h)P(D^{*c}_h)=P(D^*_h|D^{*c}_h)$$

□

With Theorem 1, we have

$$P(D^*_{t1:t2}|l{=}t1{-}1){=}\textstyle\sum_G P(D^*_{t1:t2}|l{=}t1{-}1,G)P(G) \tag{5.10}$$

$$=\textstyle\sum_G P(D^*_{t1:t2}|D^{*c}_{t1:t2},G)P(G)$$

$$=\textstyle\sum_G P(G)\int P(D^*|D^c,G,\Theta_G,T)\rho(\Theta_G|G)d\Theta_G$$

Θ_G are the parameters associated with the data $D_{t1:t2}$ corresponding to G. $\rho(\Theta_G|G)$ is the probability density function of Θ_G.

The assumption of the multinomially distributed data with the Dirichlet prior leads to the same solution (BDeu metric) of the closed form expression of the marginal likelihood $P(D_{t1:t2}|l = t1 - 1, G)$ in Equation 7.

Similarly as bayesian networks, we use MCMC to simulate $\mathcal{M}$ for $\{G\}$. Our experimental study shows that our methods in both BNs and DBNs versions output the similar results for the distributions of change-points.

5.3.2 Structure Learning of Non-stationary Bay- esian Networks

Given an observed time series data D, the structure learning problem of DBNs is equal to maximizing the posterior probability of the network structure G.

By the Bayes' rule, the posterior probability is expressed as the following:

$$P(G|D,T){=}\tfrac{P(D|G,T)P(G|T)}{P(D|T)} \tag{5.11}$$

Given a non-stationary time series data, we need to find a sequence of network structures $G^T = (G_1, \ldots, G_m)$, m segments, and a transition vector L^T, the posterior probability in

Equation 11 is replaced by Equation 12:

$$P(G^T,L^T,m|D,T)=\frac{P(D|G^T,L^T,m,T)P(G^T,L^T,m|T)}{P(D|T)} \tag{5.12}$$

$P(D|T)$ is treated as a constant, and then

$$\begin{aligned} &P(G^T,L^T,m|D,T) \\ &\propto P(D|G^T,L^T,m,T)P(G^T,L^T,m|T) \\ &\propto P(D|G^T,L^T,m,T)P(G^T|L^T,m,T)P(L^T|m,T)P(m|T) \end{aligned} \tag{5.13}$$

In the following discussion, we specify the formula for calculating each component of Equation 13.

We are using the same assumption in [227] that the networks change smoothly over time. We use the exponential priors on the change of network structures. We transform the form of the sequence of graph structures $G^T : G^T = (G_1, \ldots, G_m)$ into $G^T : G^T = (G_1, \triangle G_1, \ldots, \triangle G_{m-1})$, where $\triangle G_h : 1 \leq h \leq m-1$ is the change of edges between G_h and G_{h+1}. We calculate $P(G^T|m,T)$ as follows.

$$\begin{aligned} &P(G^T|L^T,m,T)=P(G_1,\triangle G_1,\ldots,\triangle G_{m-1}) \\ &\propto P(G_1)\textstyle\prod_{h=1}^{m-1} e^{-\lambda_s s_h} \propto P(G_1)e^{-\lambda_s \sum_{h=1}^{m-1} s_h} \propto P(G_1)e^{-\lambda_s S} \end{aligned} \tag{5.14}$$

,where $S : S = \sum_{h=1}^{m-1} s_h$, and s_h is the number of edge change between G_{h+1} and G_h. We have no prior knowledge on $P(G_1)$ and see the uniform distribution as the prior.

We assume that the data are complete and multinomially distributed with a Dirichlet prior on the parameters. We calculate $P(D_h|G_h,T)$ of each segment by following Equation

7:

$$P(D_h|G_h,T) \tag{5.15}$$

$$=\int P(D_h|G_h,\Theta_{G_h},T)\rho(\Theta_{G_h}|G_h)d\Theta_{G_h}$$

$$=\prod_{i=1}^{n}\prod_{j=1}^{q_{ih}}\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij}(I_h))}\prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}+N_{ijk})(I_h)}{\Gamma(\alpha_{ijk})}$$

We denote I_h as the segment h, Θ_{G_h} as the parameters corresponding to G_h, r_i as the number of possible values of x_i, and q_{ih} as the number of configurations of parents π_i in I_h. We let α_{ijk} and α_{ij} to be the hyperparameters for Dirichlet distributions applied in I_h. α_{ijk} is uniformly distributed inside I_h and set to $\alpha_{ijk} = \alpha/(r_i q_{ih})$. We set the equivalent sample size α equal to 1. We denote $N_{ijk}(I_h)$ as the times that x_i had value k in I_h and $N_{ij}(I_h) = \sum_{k=1}^{r_i} N_{ijk}(I_h)$.

Theorem 4. *With Theorem 1 and the Markov property, the marginal likelihood $P(D|G^T, m, T)$ is expressed as below:*

$$P(D|G^T,L^T,m,T)=\prod_{h=1}^{m}P(D_h|G_h,m,T) \tag{5.16}$$

Proof.

$$P(D|G^T,L^T,m,T)$$

$$=P(D_m^*|D_1^*,\cdots,D_{m-1}^*,G_m,m,T)\cdots P(D_1^*|G_1,m,T)$$

$$=\prod_{h=1}^{m}P(D_h^*|D_h^{*c},G_h,m,T)=\prod_{h=1}^{m}P(D_h|G_h,m,T)$$

□

With Theorem 2 and Equation 15, we get the extended BDeu metric:

$$P(D|G^T,L^T,m,T)=\prod_{h=1}^{m}P(D_h|G_h,m,T) \tag{5.17}$$

$$=\prod_{i=1}^{n}\prod_{h=1}^{m}\prod_{j=1}^{q_{ih}}\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij})+N_{ij}(I_h))}\prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}+N_{ijk}(I_h))}{\Gamma(\alpha_{ijk})}$$

We use the perfect simulation modeling to calculate the posterior probability distributions of $P(L^T|m,T)$ and $P(m|T)$. We choose the most likely m, fix the number of segments, and have $P(m|T)=1$. We use the sampling method to collect $\{G^T\}$ and will discuss the details in the subsequent section.

5.3.3 MCMC Sampling

Considering the fact that the gene expression data are usually sparse, which makes the posterior probability over structures to be diffuse [129], we choose sampling approaches rather than heuristic methods to search structural models, where a group of most likely structures could explain data better than a single one. In addition, the sampling methods also have the advantage to approximate the model space $\mathcal{M}$ for change-points simulation. We select MCMC as our sampling approach to collect G^T samples and compute the posterior probabilities of edges $\{e_{s,i,j}|1 \leq s \leq m, 1 \leq i,j \leq N\}$ in G^T. We use every single G^T sample to calculate the marginal probability $P(D_{t1:t2}|l=t1-1,G^T)$ of successive observations $y_{t1:t2}$ based on Equation 17. With a simulated sample space $\{G^T\}$ by MCMC, we get $P(D_{t1:t2}|l=t1-1)$ based on Equation 8 and further calculate the whole conditional probability distribution of change-points.

We design our algorithm based on the following considerations. First, we choose the heuristic search instead of the MCMC simulation to initialize G with only a single segment at the beginning of the algorithm. With the non-stationary nature, the data consists of multiple segments. And the possible model space and its distribution in each segment are different. In this case, MCMC may not provide a good approximation of $\mathcal{M}$ and is computationally expensive. Hence, we use the heuristic search to initializes a single G to do the perfect simulation and such change does not affect the prediction performance. In general, we take much smaller number of heuristic steps compared with MCMC, and the number of steps is proportional to the size of nodes. The detailed configurations of heuristic steps could be found in Section 4.

Second, we use KNUT move set instead of the KNKT (Known Transition Number Known Transition Time) move set [227] containing six move types, MT1-MT6 . Induced by the limitation of the initialization of a single G at the beginning, the true distributions of $P(m)$ and $P(L^T)$ are doubtful after the first round of perfect simulation. Simply using fixed change-points will distort the simulated model space. By bringing the move to shift the change-points into the move set, we allow MCMC not only to converge for L^T but also to provide a model space approximately at every time point. With this method, we improve the quality of $\mathcal{M}$ and have our algorithm converged. The procedure for our method is shown in the Algorithm as follows.

RCnsDBNs Algorithm

Input: Time series Data D, parameters p, k and λ_s
Output: $P(L^T)$, $P(m)$, and $P(\{e_{s,i,j}\})$
Begin
Use heuristic search and select a single graph G.
Run perfect simulation to sample change-points.
Calculate the distributions $P(m)$ and $P(L^T)$.
Select the most likely m and initialize G^T with G.
while $P(m)$, $P(L^T)$ and $P(\{e_{s,i,j}\})$ not converged **do**
  Run MCMC and collect the samples $\{G^T\}$.
  Simulate the change-point samples.
  Calculate the distributions $P(m)$, $P(L^T)$, and $P(\{e_{s,i,j}\})$.
  Select the most likely m and re-initialize G^T.
end while
End

5.4 Experimental Study and Evaluation

We performed all the experiments on Intel Xeon 3.2 Ghz EM64T processors with 4 GB memory. We implemented our method RCnsDBNs in Java.

We compare three approaches: 1) our approach (RCnsDBNs), 2) reversible jump Markov chain Monte Carlo Non-Stationary Dynamic Bayesian Networks (RJnsDBNs) [227], 3) Allocation Sampler Non-Stationary Dynamic Bayesian Networks (ASnsDBNs) [102, 103]. For RJnsDBNs, we use the default setting of unknown numbers and times of transitions (UNUT) in all of the data sets. RJnsDBNs is implemented in Java. ASnsDBNs is implemented in Matlab. In addition, we show the results of our method in BNs version denoted as RCnsBNs (ReCursion Non-Stationary Bayesian Networks). Both two versions of our method find very similar results on the posterior distributions of change-points. We grid-search the parameters for RCnsDBNs and RJnsDBNs for the best performance on change-point and structure estimation. For ASnsDBNs, we choose $K_{max} = 10$ for all experiments that we believe to satisfy the number of different components of the mixture vector in various data sets.

Our experimental study is based on three data sets: (i) Synthetic data set, (ii) Bone Marrow-derived Macrophages gene expression time series data (Macrophages data set), and (iii) Circadian regulation in Arabidopsis Thaliana gene expression time series data (Arabidopsis data set). We evaluate three methods from two aspects: computational performance on convergence and structure prediction accuracy.

Convergence Rate and Computational Time. ASnsDBNs with perfect simulation modeling (ASnsDBNs-PSM) [103] improves ASnsDBNs [102] on convergence. It selects parameters to give best approximation to the outputs of ASnsDBNs. Hence, we choose ASnsDBNs -PSM for computational performance comparison. We follow Grzegorczyk's work in [103] and evaluate ASnsDBNs-PSM and our method with the proportion of edges denoted by η for which potential scale reduction factors (PSRFs) [88] lies below the pre-defined threshold. PSRFs=1 shows perfect convergence and that PSRFs<1.1 is seen as the sufficient condition for convergence [103, 88]. $0 \leq \eta \leq 1$ and higher η values indicate better

convergence.

RJnsDBNs does not output graph samples. We use the variation of edge posterior probabilities (VEPP) to measure the convergence of its output. $VEPP = \frac{1}{m \cdot N \cdot N} \sum_{s=1}^{m} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{|P(e_{s,i,j}^{I+\Delta I}) - P(e_{s,i,j}^{I})|}{P(e_{s,i,j}^{I})}$, where I is the number of iterations continuously sampling in MCMC, and $P(e_{s,i,j}^{I})$ is the posterior probability of an edge $e_{i,j}$ in the graph G_s that dominates the sth segment computed from I iterations. Once MCMC converges, $|P(e_{s,i,j}^{I+\Delta I}) - P(e_{s,i,j}^{I})| \to 0$ with $I \to +\infty$. Hence, VEPP values close to 0 indicate that a MCMC chain converges to a stationary distribution. We use a pre-defined threshold σ. When $VEPP < \sigma$, we decide that MCMC converges and calculate the computational time.

Structure Prediction Accuracy. To compare the inferred structure results from different data sets, we follow the evaluation method introduced in [129, 262, 102]. For the synthetic data set, we compare the inferred network structures with the true networks. For each real data set, we first collect gold standard reference networks as the ground truth. For the Macrophages data set, such reference networks are available in [148, 224, 102]. For the Arabidopsis data set, we collect the network information from [183, 228, 62, 208]. In case where we have ground truth network structure (the Bone Marrow data set and Arabidopsis data set), we use the area under receiver operating characteristic curve (AUROC) values to evaluate the performance. In addition, for each data set, we show the posterior distribution of the number of segments and the locations of change-points. Before we discuss the details of experimental results, we present the characteristics of our data set first below.

5.4.1 Data Sets

We evaluate our method RCnsDBNs on a synthetic data and two gene expression data sets used in [227, 102]. We preprocess the original gene expression data sets by following Zhao's work [283]. We set the values of a missed time point with the mean of its two neighbors; i.e., $X_{i,t} = (X_{i,t-1} + X_{i,t+1})/2$ if $1 < t < T$. If the missed values are at the beginning or end, simply set the same value as its neighbor; i.e., $X_{i,t} = X_{i,t+1}$ if $t = 1$ or $X_{i,t} = X_{i,t-1}$ if $t = T$.

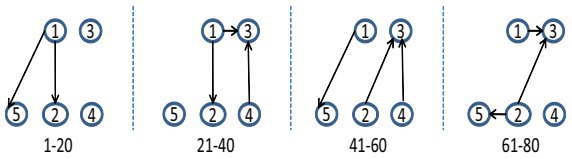


Figure 5.1: The synthetic networks.

In the following, we show the details of each data set.

Synthetic Data. We created a synthetic time series data with 80 time points and binary valued observations. It was generated by a sequence of 5 node networks with 3-4 edge changes between successive segments. The change-points of the graph structures happened at times 20, 40, 60. We showed the true networks in Figure 5.1.

Bone Marrow-derived Macrophages Gene Expression Data. We use the Macrophage data sets previously investigated in [102]. The data sets contain three genes, *Irf1*, *Irf2* and *Irf3*, related to Interferon regulatory factors (IRFs), proteins central to the mammalian innate immunity [120, 224]. The Macrophage data sets were sampled from different conditions: (I) Infection with Cytomegalovirus (CMV), (II) Treatment with Interferon Gamma (IFN_γ), and (III) Infection with Cytomegalovirus after pretreatment with IFN_γ $(CMV + IFN_\gamma)$. Each data set has 25 time points collected with the interval 30 minutes. We follow Grzegorczyk's work [102] and use $Irf2 \leftrightarrow Irf1 \leftarrow Irf3$ as the gold standard. We assume that the network is invariant over time.

Arabidopsis Thaliana Circadian Regulation Gene Expression Data. We use the Arabidopsis Thaliana Circadian data investigated in [102]. The data sets consist of 9 genes, *LHY*, *CCA1*, *TOC1*, *ELF4*, *ELF3*, *GI*, *PRR9*, *PRR5*, and *PRR3*. The group of genes create transcriptional feedback loops and are critical to understand the internal clock-signalling network of plant. The Arabidopsis data are sampled from two light-dark conditions: (I) 10h:10h light/dark cycle and (II) 14h:14h light/dark cycle. Each data set contains 13 time points collected with the interval of 2 hours. We build a gold standard network based on the biological literatures [183, 228, 62, 208, 110, 190]. In this network, CCA1 and LHY proteins directly bind to the promoter of TOC1 to represses the expression of TOC1. The

pseudo-response regulators PRR5 and PRR9 are activated by CCA1 and LHY and repress CCA1 and LHY subsequently. G1 improves the expression of TOC1. ELF4 is repressed by CCA1. For a detailed referred graph figure, please refer to our previous work [141].

5.4.2 Convergence and Computational Performance

We first compared the computational performance between our method RCnsDBNs and ASnsDBNs-PSM. The curves of fraction of edges with PSRFs<1.04 on two methods for Thaliana T20 data is showed in Figure 5.2 and the VEPP curves in Figure 5.3. We calculated the PSRFs and VEPP scores from 10 independent MCMC chains. We found that RCnsDBNs and ASnsDBNs-PSM have the similar convergence rate measured in terms of MCMC sampling iterations. However, for 250,000 iterations, it takes ASnsDBNs-PSM more than **350 hours** while RCnsDBNs only needs less than **1 minute**. Even considering the fact that two algorithms are implemented in different programming languages (RCnsDBNs in java and ASnsDBNs-PSM in Matlab), compared with ASnsDBNs-PSM, our method has much better computational efficiency.

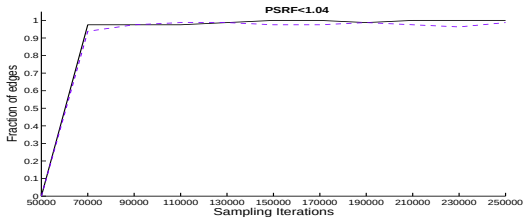


Figure 5.2: The curves of fraction of edges with PSRFs<1.04 on RCnsDBNs and ASnsDBNs-PSM for Thaliana T20 data. RCnsDBNs: black solid line ; ASnsDBNs-PSM: blue dashed line.

For the comparison between RJnsDBNs and our approach, we set $\sigma = 0.05$ for the convergence of VEPP values and listed the number of iterations and computational time in Table 5.1. In multiple data sets, RCnsDBNs converges much faster than RJnsDBNs. Compared with RJnsDBNs , RCnsDBNs got 6 folds computational improvement on CMV data, 6 folds on $CMV + IFN_\gamma$ data, 9 folds on IFN_γ data. On Arobidopsis microarray

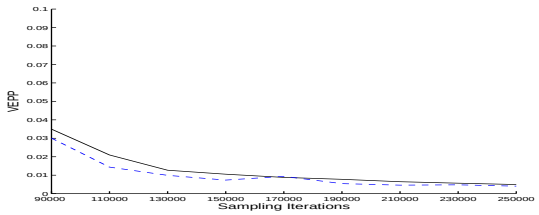


Figure 5.3: The VEPP curves on RCnsDBNs and ASnsDBNs-PSM for Thaliana T20 data. RCnsDBNs: black solid line ; ASnsDBNs-PSM: blue dashed line.

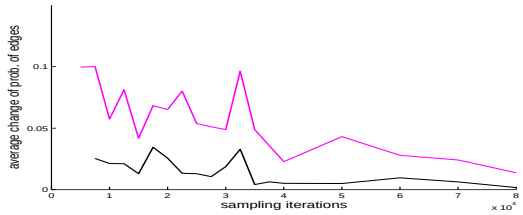


Figure 5.4: The VEPP curves for CMV data. RCnsDBNs ($p = 0.01, \lambda_s = 2$): black solid line ; RJnsDBNs ($\lambda_m = 0.65, \lambda_s = 2$): magenta dash-dot line.

Table 5.1: The comparison of computational performance

N	RJnsDBNs	I	C_T	RCnsDBNs	I	T	I	C_T	speedup
5	*Synthetic Data* $(\lambda_m =, \lambda_s = 2)$	$500,000$	$34.28m$	*Synthetic Data*	$500,000$	$17.91m$	$400,000$	$14.36m$	2.39
3	*CMV* $(\lambda_m = 0.65, \lambda_s = 2)$	$50,000$	$7.316s$	*CMV*	$50,000$	$4.333s$	$10,000$	$1.127s$	6.49
3	IFN_γ $(\lambda_m = 0.001, \lambda_s = 2)$	$80,000$	15.962	IFN_γ	$80,000$	$6.615s$	$17,500$	$1.646s$	9.70
3	$CMV + IFN_\gamma$ $(\lambda_m = 1, \lambda_s = 2)$	$130,000$	$26.152s$	$CMV + IFN_\gamma$	$130,000$	$16.223s$	$25,000$	$4.029s$	6.49
9	*ArobidopsisT20* $(\lambda_m = 0.0005, \lambda_s = 2)$	$30,000$	$6.274s$	*ArobidopsisT20*	$50,000$	$5.363s$	$100,000$	$10.314s$	0.61
9	*ArobidopsisT28* $(\lambda_m = 0.005, \lambda_s = 2)$	$30,000$	$6.048s$	*ArobidopsisT28*	$50,000$	$15.26s$	$100,000$	$28.325s$	0.21

N is the number of genes in the data sets. I is the number of iterations. T is the computational time. C_T is the computational time for convergence. $\sigma = 0.05$ is used to decide the convergence of the results.

data, RJnsDBNs took less time than RCnsDBNs. However, it failed to detect any meaningful change-point as RCnsDBNs and ASnsDBNs did on Arobidopsis data. In addition, we showed the VEPP curves of two approaches for CMV data in Figure 5.4.

5.4.3 Stability of Results

We use $k = 1$ for all the experiments because the gene expression data usually has limited time points and larger k values eliminate short segments. The value of parameter p is adjusted for the purpose of the convergence of results for different dominant segment numbers m. We grid-search the values of p between $0.00001 \sim 0.5$ for the effective range on the preferred segmentation. For the synthetic data, it has four segments ($m = 4$). For three Macrophages data, we selected $m = 1$ based on the assumption of a single IRFs network structure with varying parameters [102]. For Arobidopsis T20 data, most of the p range leads to $m = 1$. Finally, for Arobidopsis T28 data, we chose $m = 2$ with the consideration of the external light/dark cycle condition.

Table 5.2: The effective range of parameter p for RCnsDBNs

	effective parameter range of p
Synthetic (m=4)	$0.02 \sim 0.032$
CMV (m=1)	≤ 0.009
CMV+IFN$_\gamma$ (m=1)	≤ 0.0006
IFN$_\gamma$ (m=1)	≤ 0.0001
Arobidopsis T20 (m=1)	≤ 0.5
Arobidopsis T28 (m=2)	$0.18 \sim 0.23$

5.4.4 Structure Prediction and Change-point Detection

In the following, we will show the results of predicted structures and detected change-points.

The results on synthetic data. We compared two discrete models, RCnsDBNs and RJnsDBNs, on synthetic data. RCnsDBNs totally runs 16 rounds to get converged, and each round uses 5,000 iterations for burn-in and then takes additional 20,000 iterations to collect samples ; RJnsDBNs runs 100,000 iterations for burn-in and then takes additional 400,000 iterations to collect samples. RCnsDBNs initializes G with additional 1000 heuristic search steps.

We showed the predicted posterior distributions on the numbers of segments and change-

Table 5.3: The AUROC values of RCnsDBNs on synthetic data

	Synthetic Data
RCnsBNs	$G1:1; G2:0.6078;$ $G3:0.4706; G4:0.6078$
RCnsDBNs (equivalence class considered)	$G1:0.9688; G2:0.6719;$ $G3:0.5938; G4:0.6406$

TP, true positive; FP, false positive; TN, true negative; FN, false negative.
Sensitivity = TP/(TP+FN).
Specificity = TN/(TN+FP).
Complementary Specificity = 1- Specificity = FP/(TN+FP).
The ROC curves are plotted with the Sensitivity scores against the corresponding Complementary Specificity scores.

points in Figure 5.5. RCnsDBNs correctly identified 4 segments and its predicted change-points are close to the true times at 20, 40, and 60 while RJnsDBNs failed to identify meaningful change-points. The AUROC scores of predicted structures by RCnsDBNs is showed in Table 5.3. When the equivalence class of bayesian network structures [53] were considered, the AUROC scores of all segments were increased, which were shown in the same table.

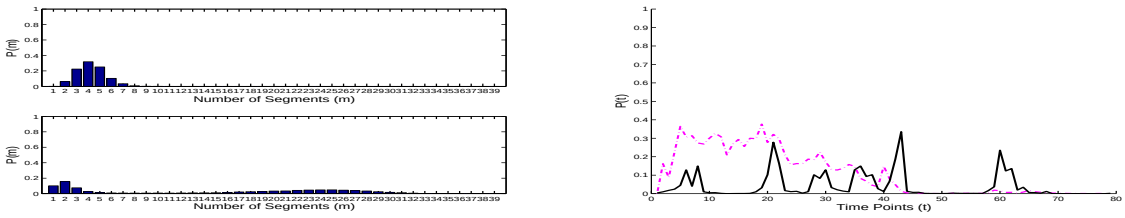


Figure 5.5: Comparison of two methods on the synthetic data. Up: The posterior probabilities of the numbers of segments P(m) (top: RCnsDBNs ($p = 0.031, \lambda_s = 0.5$); bottom: RJnsDBNs ($\lambda_m = 0.4, \lambda_s = 0.5$)). Low: The posterior probabilities of the change points P(t) (RCnsDBNs: black solid line; RJnsDBNs: magenta dash-dot line).

The results on Macrophages data. On the CMV Macrophages data, RCnsDBNs totally runs 4 rounds to get converged, and each round uses 500 iterations for burn-in and then takes additional 2,000 iterations to collect samples ; RJnsDBNs runs 10,000 iterations for burn-in and then takes additional 40,000 iterations to collect samples. On the $CMV + IFN_\gamma$ Macrophages data, RCnsDBNs totally runs 5 rounds to get converged, and each round uses 1,000 iterations for burn-in and then takes additional 4,000 iterations to collect samples

; RJnsDBNs runs 26,000 iterations for burn-in and then take additional 104,000 iterations to collect samples. On the IFN_γ Macrophages data, RCnsDBNs totally runs 7 rounds to get converged, and each round uses 500 iterations for burn-in and then takes additional 2,000 iterations to collect samples ; RJnsDBNs runs 16,000 iterations for burn-in and then take additional 64,000 iterations to collect samples. In both data sets, ASnsDBNs runs 10,000 iterations for burn-in and then take additional 40,000 iterations to collect samples. RCnsDBNs initializes G with additional 100 heuristic search steps.

In Figure 5.6, 5.7, and 5.8, we show the posterior probabilities of the numbers of segments and change-points on Macrophages data sets.

For the CMV data, we observe that ASnsDBNs clearly identifies a dominant 3-segment in the data set while the posterior probabilities produced by RJnsDBNs are almost flat. There is a consensus among three methods that the most probable change-point occurs around the location 5. The results of three methods are consistent with the biological phenomenon that the simultaneous responses of Macrophages happen under the attack of Cytomegalovirus [102]. In order to assess the network prediction performance, we show the AUROC scores in Table 2. We find that all methods perform well in the CMV data with the AUROC scores equal to 1.

For the $CMV+IFN_\gamma$ data, both RJnsDBNs and ASnsDBNs methods identify 1 segment, which [102] explained as a coexistence state between virus and its host cell [22, 102]. And their posterior probabilities are flat. Different from these two methods, RCnsDBNs found two posterior peaks at 3 and 8. Such finding indicates the coexistence state may not happen at the beginning under both the IFN_γ treatment and invasion of virus. In Table 4, we find that RCnsDBNs and ASnsDBNs show a much better network prediction with the AUROC score equal to 0.6667 while in RJnsDBNs the AUROC score is equal to 0.2222.

For the IFN_γ data, there is a postulated transition with the immune activation under the treatment of IFN_γ. Both RJnsDBNs and ASnsDBNs infer 1 segments. RJnsDBNs and ASnsDBNs identify a same posterior peak at the location around 5. RCnsDBNs finds two

posterior peaks of transition time at 9 and 13. On the assessment of the predicted network structures, the AUROC scores are 0.7778 in RCnsDBNs and RJnsDBNs, and 0.6667 in ASnsDBNs.

For each Macrophages data set using RCnsDBNs and RJnsDBNs methods, we find that the posterior probability distributions of any edge do not change much across different segments. This finding is consistent with the assumption that the underlying network structure does not change through the time.

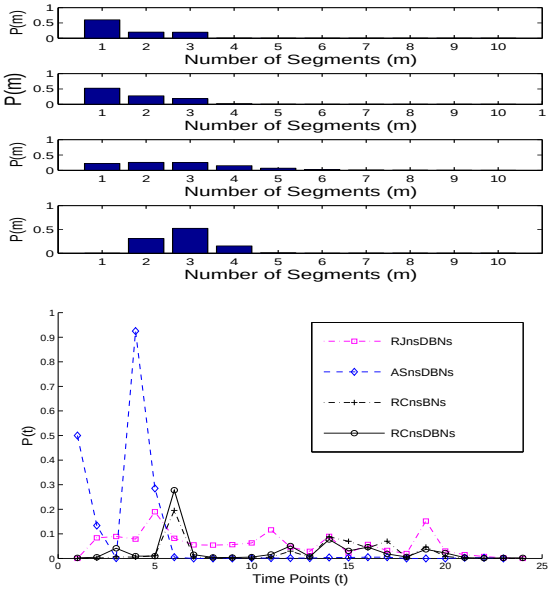


Figure 5.6: Comparison of four methods on CMV Macrophage data. Up: The posterior probabilities of the numbers of segments P(m) (from the top to the bottom: RCnsBNs ($\lambda_s = 2$), RCnsDBNs ($\lambda_s = 2$), RJnsDBNs ($\lambda_m = 0.65, \lambda_s = 2$), and ASnsDBNs). Low: The posterior probabilities of the change points P(t).

The results on Arabidopsis data. On the Arabidopsis data, RCnsDBNs totally runs 10 rounds to get converged, and each round uses 5,000 iterations for burn-in and

Table 5.4: Comparison of AUROC values on Macrophage data

	CMV	IFN_γ	$CMV + IFN_\gamma$
RJnsDBNs	1	0.7778	0.2222
ASnsDBNs	1	0.6667	0.6667
RCnsBNs	1	0.5556	0.6667
RCnsDBNs	1	0.7778	0.6667

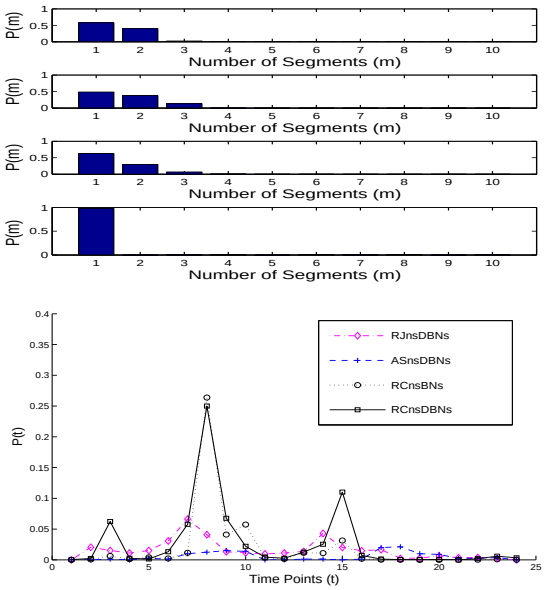


Figure 5.7: Comparison of four methods on $CMV + IFN_\gamma$ Macrophage data. Up: The posterior probabilities of the numbers of segments P(m) (from the top to the bottom: RCnsBNs ($\lambda_s = 2$), RCnsDBNs ($\lambda_s = 2$), RJnsDBNs ($\lambda_m = 1, \lambda_s = 2$), and ASnsDBNs). Low: The posterior probabilities of the change points P(t).

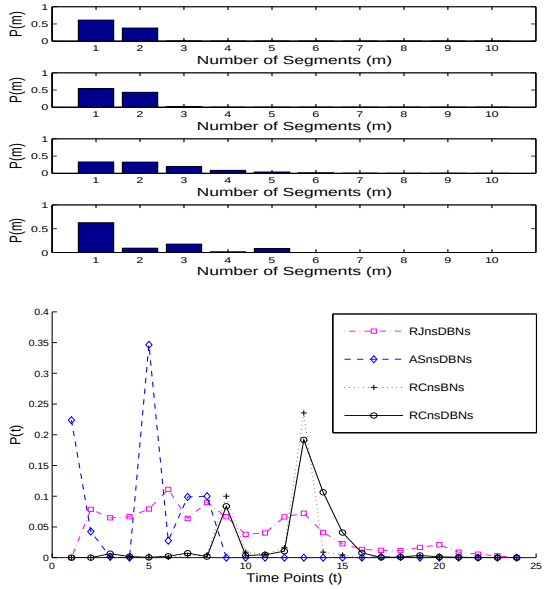


Figure 5.8: Comparison of four methods on IFN_γ Macrophage data. Up: The posterior probabilities of the numbers of segments P(m) (from the top to the bottom: RCnsBNs ($\lambda_s = 2$), RCnsDBNs ($\lambda_s = 2$), RJnsDBNs ($\lambda_m = 0.001, \lambda_s = 2$), and ASnsDBNs). Low: The posterior probabilities of the change points P(t).

then takes additional 20,000 iterations to collect samples ; RJnsDBNs runs 6,000 iterations for burn-in and then take additional 24,000 iterations to collect samples; ASnsDBNs runs 990,000 iterations for burn-in and then take additional 10,000 iterations to collect samples. Considering larger size of variables and thereafter the larger model space compared with other two data sets, RCnsDBNs takes more heuristic search steps and initializes G with additional 10000 heuristic iterations.

In Figure 5.9 and 5.10, we show the posterior distributions of the numbers of segments and changepoints on two Arabidopsis data sets. For the Arabidopsis T20 data, the dominant samples in RJnsDBNs and ASnsDBNs are respectively 2 and 3 segments. For the Arabidopsis T28 data, RJnsDBNs infers 1 segment and ASnsDBNs infers 5 segments. In both data sets, we find that the difference between the posterior peaks of changepoints and the time points nearby in RJnsDBNs are not noticeable. Hence, for this data set, we only use a single network in RJnsDBNs to compare with other methods. Using ASnsDBNs, the posterior peaks of change-points on T20 data are 1, 5 and those on T28 are 2, 7, 10. In [102], the results of ASnsDBNs are explained as a phase shift incurred by different dark/light cycles. Our method RCnsDBNs had the same finding by identifying the peaks at 5, 7, and 10 on T20 data and the peaks at 2, 6, and 9 on T28 data. And in addition, RCnsdBNs finds a peak around 10 on T20 data. This time point is exactly the beginning of the new light/dark cycle.

We evaluated the network reconstruction accuracy of three methods by comparing with the reference network showed in Section 6.2.1. We show the AUROC scores in Table 5.5. Our method outperforms RJnsDBNs in both datasets and has competitive performance on structure prediction accuracy against ASnsDBNs.

Table 5.5: Comparison of AUROC values on Arabidopsis data

	Arabidopsis T20	Arabidopsis T28
RJnsDBNs	0.5035	0.3893
ASnsDBNs	0.5929	0.5641
RCnsBNs	$G1 : 0.4856$	$G1 : 0.4856; G2 : 0.5315$
RCnsDBNs	$G1 : 0.5183$	$G1 : 0.5925; G2 : 0.5979$

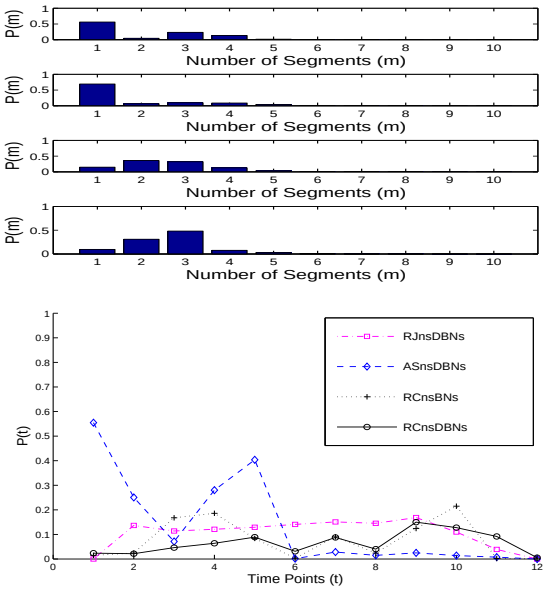


Figure 5.9: Comparison of four methods on Arabidopsis T20 data. Up: The posterior probabilities of the numbers of segments P(m) (from the top to the bottom: RCnsBNs ($\lambda_s = 2$), RCnsDBNs ($\lambda_s = 2$), RJnsDBNs ($\lambda_m = 0.0005, \lambda_s = 2$), and ASnsDBNs). Low: The posterior probabilities of the change points P(t).

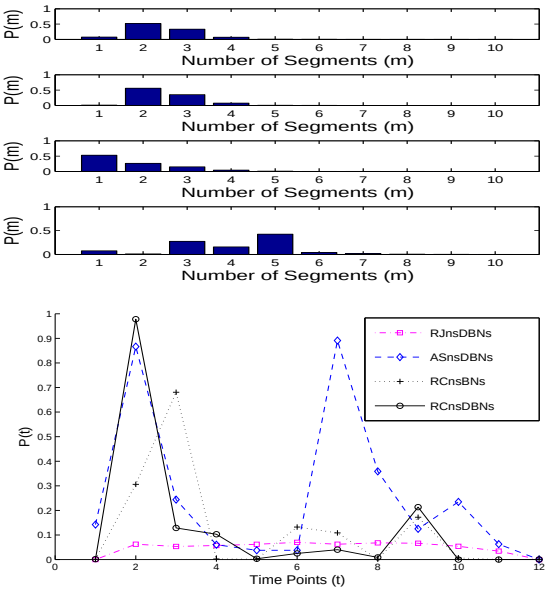


Figure 5.10: Comparison of four methods on Arabidopsis T28 data. Up: The posterior probabilities of the numbers of segments P(m) (from the top to the bottom: RCnsBNs ($\lambda_s = 2$), RCnsDBNs ($\lambda_s = 2$), RJnsDBNs ($\lambda_m = 0.005, \lambda_s = 2$), and ASnsDBNs). Low: The posterior probabilities of the change points P(t).

Chapter 6

Preliminary Work (IV): Bayesian Network Structure Learning with Text Priors

In this chapter, we will present a new Bayesian Network structure learning method with text priors that are encoded by non-parametric hierarchical topic trees. Our method retrieved structure prior information from unstructured text data. And different from the existing reverse engineering methods with text priors [175, 243, 86], our method does not need standard glossary or ontology systems.

6.1 Methods

6.1.1 Structure Inference of Bayesian Networks

Bayesian networks (BNs) are a special case of probabilistic graphic models. A static BN is defined by a graph structure G, and a complete joint probability distributions of its nodes $P(X) = P(X_1, \ldots, X_n)$. The structure $G : G = (X, E)$ is an directed acyclic graph (DAG), which contains a set of variables $X = \{X_1, \ldots, X_n\}$, and a set of directed edges E, which

define the causal relations between variables. Since the graph structures of static BNs are directed acyclic, the joint distributions can be decomposed as $P(X_1, \ldots, X_n) = \prod_i P(X_i|\pi_i)$, where π_i is the parents of the node (variable) X_i.

Given an observed data x, the structure inference problem of BNs is equal to maximize the posterior probability of the network structure G. By the Bayes' rule, the posterior probability is expressed as following:

$$P(G|x) = \frac{P(x|G)P(G)}{P(x)}. \tag{6.1}$$

When $P(x)$ is treated as a constant, Equation 6.1 further extends to

$$P(G|x) \propto P(x|G)P(G). \tag{6.2}$$

We have no prior knowledge on $P(G)$ and see the uniform distribution as the prior.

In this paper, we assume that the observations are independent, complete and multinomially distributed with a Dirichlet prior on the parameters Θ_G, we have the BDeu [117] solution to $P(x|G)$:

$$\begin{aligned} P(x|G) &= \int f(x|G, \Theta_G)\rho(\Theta_G)d\Theta_G \\ &= \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})}. \end{aligned} \tag{6.3}$$

Θ_G are the parameters associated with the data x corresponding to G. $\rho(\Theta_G)$ is the probability density function of Θ_G. r_i is the number of possible discrete values of X_i. q_i is the number of configurations of parents π_i for the variable x_i. N_{ijk} is the times that X_i had value k. $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. α_{ijk} and α_{ij} are the hyperparameters for Dirichlet distribution. α_{ijk} is assumed to be uniformly distributed and is set to $\alpha_{ijk} = \alpha/(r_i q_i)$. The equivalent sample size α is set to 1.

6.1.2 Hierarchical Latent Dirichlet Allocation Modeling

In this paper, we used hierarchical Latent Dirichlet Allocation (hLDA) [26, 25] to model the hierarchical tree topology of topics of a corpus of documents $D = \{d_1, \cdots, d_n\}$. A document $d_i : 1 \leq i \leq n$ is a sequence of words denoted as $d_i =< w_1, \cdots, w_{m_i} >$. A topic in hLDA is defined as a word distribution over a vocabulary V. V is a lexicon set containing all the words in a document corpus D. A hLDA model consists of two key components: nested Chinese Restaurant Process (nCRP) and Dirichlet Mixture Model. We will discuss these two techniques in the following subsections.

6.1.2.1 Nested Chinese Restaurant Process

Chinese Restaurant Process (CRP) is a special case of Dirichlet Process of latent mixture component in Finite Dirichlet Process Mixture Model (DPMM) when the number of mixture components K goes infinity [197]. It is a distribution with a single real value parameter γ over partitions of integers. Suppose that there exists a restaurant with infinite number of tables, each of which can accommodate infinite customers. For a group of n customers lined as a sequence $\{s_1, \cdots, s_n\}$, the first customer chooses the first table t_1 with the probability 1 and the nth customer selects one of occupied tables $t_i : 1 \leq i \leq K$ or a new unoccupied table t_{K+1} based on the distribution as follows.

$$\begin{aligned} P(t_i|s_1, \cdots, s_{n-1}) &= \frac{n_i}{n-1+\gamma} \\ P(t_{K+1}|s_1, \cdots, s_{n-1}) &= \frac{\gamma}{n-1+\gamma}, \end{aligned} \tag{6.4}$$

where n_i is the number of customers sitting at table t_i and $\sum_i^K n_i = n - 1$.

Nested Chinese Restaurant Process (nCRP) is a model that extends CRP by incorporating hierarchical tree topology to handle inherent structures of data. In nCRP model, we have a tree structure T that could have infinite branches and an infinite depth. Every node in the tree can be seen as a restaurant described in CRP. Each edge from a parent node r_p

to a child node r_c in the tree represents a direct connection between a table t_i in r_p and r_c, that is, that the customers sitting at the table t_i in r_p will visit r_c. We follow the work in [26] and have the depth of the tree fixed as L. Hence, every customer is exposed to a set of sequences of restaurants, which are all the existing pathes with the depth l from the root to the leafs in the tree or potential new branches that extend from existing internal nodes to the depth l. Hence, each node r in the tree represents a path c_r that is a sequence of nodes ordered from root to r. We denote the c_r of a node r with the level $l(r)$ as $A(r) =< a_i >$ where $i : 1 \leq i \leq l(r)$ is the level. The probability that a customer s_n chooses a path c_r is

$$P(c_r|s_1, \cdots, s_{n-1}, T) = \frac{\gamma}{a_{l(r)} + \gamma} \prod_{i=2}^{l(r)} \frac{n(a_i)}{n(a_{i-1}) + \gamma}, \tag{6.5}$$

where $n(a_i)$ is the number of customers that visited the restaurant a_i.

In hLDA model, each document is treated as a customer and each topic is seen as a restaurant in a nCRP tree. Hence nCRP provides a prior distribution of the document assignment over sequences of topics.

6.1.2.2 Dirichlet Process Mixture Model

Suppose we have observations $Y = \{y_1, \cdots, y_n\}$ and $y_i : 1 \leq i \leq n$ is independently drawn from an unknown distribution F. The general procedure of Dirichlet Process Mixture Model [197] is as follows.

$$\begin{aligned} y_i &\sim F(\theta_c) \\ \theta_{c_i} &\sim H(c_i) \\ c_i &\sim DirichletProcess(\gamma). \end{aligned} \tag{6.6}$$

In hLDA model, $Y = D$ and nCRP serves as the Dirichlet Process to generate the mixture class. Given a chosen nCRP path c, the generative procedure to draw a single document d consists of two steps. For each word position i in d, first we select a level l in the path c;

second, we draw a word from the topic r at level l in c. Both steps that data are generated from the discrete distribution with a Dirichlet prior. Hence, we have the new form of the Dirichlet Process Mixture Model in hLDA model as follows.

Algorithm 1 The Generative Process to Generate a Document

PROCEDURE: sample a document
$c_i \sim DirichletProcess(\gamma)$
for each word w **do**
  $\theta_l \sim Dirichlet(\alpha)$
  $l \sim Discrete(\theta_l)$
  $\theta_w \sim Dirichlet(\eta)$
  $w \sim Discrete(\theta_w)$
end for

6.1.3 Bayesian Network Structure Inference with Text Priors

We chose Markov chain Monte Carlo (MCMC) to sample the network and hierarchical tree structures from the posterior $P(G|x,T)$ and $P(T|D)$. We used Gibbs sampling to simulate $P(T|D)$ and employed Metropolis Hasting algorithm to sample G from $P(G|x,T)$. We proposed two optional strategies to mix two sampling sub-procedures.

Algorithm 2 Option (I): Bi-way Mixing

PROCEDURE: iterative sampling $\{G,T\}$
while $i < maximum\ \#iterations$ **do**
  sample G depending on T
  sample T depending on G
end while

Algorithm 3 Option (II): One-way Mixing

PROCEDURE: iterative sampling $\{G,T\}$
while $i < maximum\ \#iterations$ **do**
  sample T
  sample G depending on T
end while

By considering that in our applications, documents summarized by experts may contain much less noises than the observations to show a different view of network structures, in this

paper we chose the second strategy to do the one-way mixing. We called our mixing strategy as *text regularity*.

We reformulated Equation 6.1 as follows.

$$\begin{aligned} P(G,T|x,D) &= P(G|T,D,x)P(T|D,x) \\ &= P(G|T,x)P(T|D) \\ &= \frac{P(x|G,T)P(G|T)}{P(x|T)} \cdot \frac{P(D|T)P(T)}{P(D)} \\ &= \frac{P(x|G)P(G|T)}{P(x)} \cdot \frac{P(D|T)P(T)}{P(D)}, \end{aligned} \tag{6.7}$$

and have

$$P(G,T|x,D) \propto P(x|G)P(G|T)P(D|T)P(T) \tag{6.8}$$

We set a uniform distribution for $P(T)$. To calculate $P(D|T)$, we followed the generative procedure in Algorithm 1. For $P(x|G)$, we applied BDeu metric in Equation 6.3. To calculate $P(G|T)$, we used an additional structure that is a symmetric matrix M with the dimension n. We first acquired the corresponding undirected graph G' of a DAG G by eliminating the direction of edges in G. For each pair of indices (i,j) in M_b, if the distance between nodes v_i and v_j is less than τ in G', we set $M_{i,j} = 1$. In this paper, we used $\tau = 2$. It could be seen as an extension of Markov Blanket [212]. Given a node v_i, instead of defining a set MB that contains v_i's parents, children, and children's other parents, we define another set MB' by adding to MB with parents' children, parents' parent, and children's children. Suppose that nodes v_i and v_j assigned pathes c_i and c_j. We denoted the number of topics in a path as $|c|$ and the shared topics between two pathes c_i and c_j as $|c_i \cap c_j|$. We have the probability that $P(M_{i,j} = 1|T) = \frac{|c_i \cap c_j|}{|c|}$ where $|c_i| = |c_j| = |c|$. For each entry in the matrix M, there exists a Bernoulli distribution with the parameter $P(M_{i,j} = 1|T)$. Since M is symmetric, we only

need to consider the elements below the diagonal. Hence, we have

$$P(G|T) = \prod_{i,j=1,i<j}^{i,j=n} P(M_{i,j}|T). \tag{6.9}$$

In the following, we will discuss two sampling steps respectively.

6.1.3.1 Sampling of Topic Trees.

We followed Blei's work [26, 25] to divided the Gibbs sampling into two steps: sampling pathes of topics for documents and sampling levels in pathes for words.

We denoted all the words in a document corpus D as w, the words of a document d in D as w_d, the words in the documents of $D - \{d\}$ as w_{-d}, the words in D excluding the nth word in a document d as $w_{-(d,n)}$, a single word in d as ω, all the assigned levels in pathes for w as z, the levels for w_d as z_d, the levels for w_{-d} as z_{-d}, all the pathes for D as c, the path for d as c_d, and the pathes for $D - \{d\}$ as c_{-d}

Path Sampling. With the fixed depth l of the topic tree T and all the word levels z, we have

$$P(c_d|w, z, c_{-d}) \propto P(c_d|c_{-d})P(w_d|c, z, w_{-d}). \tag{6.10}$$

We calculated $P(c_d|c_{-d})$ with Equation 6.5. In Algorithm 1, we see a discrete distribution as a special case of a multinomial. Given the assumption of multinomial distribution of lexicon words with the Dirichlet prior parameterized by η, we have the distribution of potential pathes for a single document d as

$$\begin{aligned} P(w_d|c, z, w_{-d}) = \ & \prod_{i=1}^{l} \frac{\Gamma(\sum_{\omega} \#[z_{-d}{=}i, c_{-d,i}{=}c_{d,l}, w_{-d}{=}\omega] + |V|\eta)}{\prod_{\omega} \Gamma(\#[z_{-d}{=}i, c_{-d,i}{=}c_{d,l}, w_{-d}{=}\omega] + \eta)} \\ & \frac{\prod_{\omega} \Gamma(\#[z{=}i, c_i{=}c_{d,l}, w{=}\omega] + |V|\eta)}{\Gamma(\sum_{\omega} \#[z{=}i, c_i{=}c_{d,l}, w{=}\omega] + \eta)}. \end{aligned} \tag{6.11}$$

Word Level Sampling. With all the pathes c of documents D, we have the distribution

of level allocations for each word as

$$P(z_{d,n}|z_{-(d,n)}, c, w) \propto P(z_{d,n}|z_{d,-n})P(w_{d,n}|c, z, w_{-(d,n)}). \tag{6.12}$$

With the same assumption of multinomial distributions with Dirichlet priors on words along levels and over topics, respectively parameterized by α and η, we have

$$P(z_{d,n}|z_{d,-n}) \propto \alpha + \#[z_{-(d,n)} = z_{d,n}], \tag{6.13}$$

and

$$\begin{aligned} P(w_{d,n}|c, z, w_{-(d,n)}) \qquad &\propto \eta + \#[z_{-(d,n)} = z_{d,n}, \\ &c_{z_{d,n}} = c_{d,z_{d,n}}, w_{-(d,n)} = w_{d,n}]. \end{aligned} \tag{6.14}$$

6.1.3.2 Sampling of Bayesian Network Structures.

We used the general Metropolis Hasting algorithm with three move types [116]: (i) add an edge; (ii) remove an edge; (iii) reverse an edge. Instead of using the directed acyclic graph G as the state, we chose the symmetric matrix M as its equivalence. We saw the state transition procedure of M as two steps: (i) to randomly select multiple entries M_{ij}s; (ii) to sample the values of M_{ij}s with Bernoulli distributions. Hence, we proposed the acceptance criterion as follows.

$$P(M, M') = \min\{1, \frac{P(M'|x)P(M|M', T)}{P(M|x)P(M'|M, T)}\}, \tag{6.15}$$

where $P(M|M', T)$ is calculated with Equation 6.9 and the ratio $\frac{P(M|M',T)}{P(M'|M,T)}$ has the form

$$\frac{P(M|M', T)}{P(M'|M, T)} = \prod_{i<j, P(M_{i,j}) \neq P(M'_{i,j})} \frac{P(M_{i,j}|T)}{P(M_{i,j}|T)}. \tag{6.16}$$

We used one of the discussed move types to propose new sample G' and approximated $P(G|G')$ with $P(M|M')$. When G' is a cyclic graph, we reject the new state M'.

6.2 Experimental Study and Evaluation

We performed all the experiments on Intel Xeon 3.2 Ghz EM64T processors with 4 GB memory. We implemented our method in Java.

We compare two approaches: 1) our approach denoted as BNsTP, 2) the general bayesian network structure learning algorithm denoted as BNs [116] Our experimental study is based on three data sets: two hybrid educational data set and one biological microarray gene expression data. We evaluate three methods from structure prediction accuracy.

Structure Prediction Accuracy. To compare the inferred structure results from different data sets, we follow the evaluation method introduced in [129, 262, 102]. For the hybrid data sets, we compare the inferred network structures with the true networks. For the real biological data sets, we collect gold standard reference networks as the ground truth. In case where we have ground truth network structure (the hybrid synthetic data set), we use the area under receiver operating characteristic curve (AUROC) values to evaluate the performance. In case where we have no true network structure (the biological data set), we compare our findings with other researchers' work. Before we discuss the details of experimental results, we present the characteristics of our data set first below.

6.2.1 Data Sets

Synthetic Educational Data.

The original data set is from the Dynamic Learning Maps Alternate Assessment System (DLM) project in the the Center for Educational Testing and Evaluation (CETE) at the University of Kansas [2]. It represents an example of a small portion of a learning map, which is one of the results in an effort of building an alternate assessment system for students with

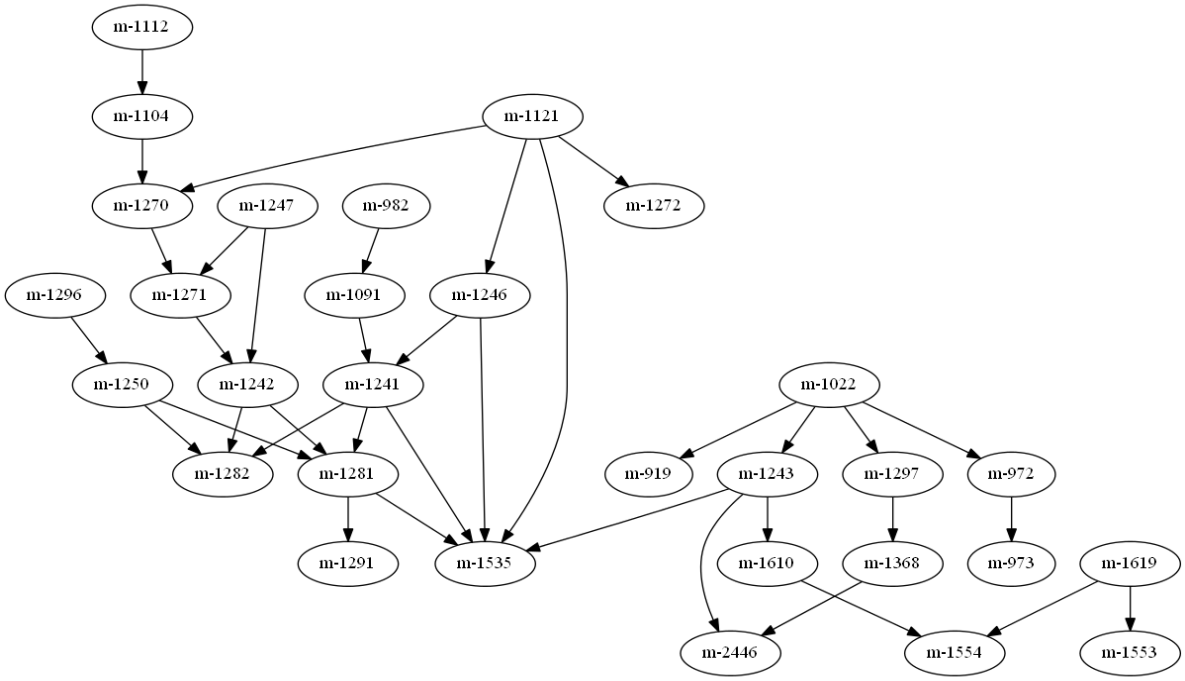


Figure 6.1: The Math Learning Pathways with 30 Nodes

significant cognitive disabilities in educational testing field. The map is designed to show what students know in ways that traditional multiple-choice tests cannot. This particular portion of the map has 30 nodes, each of which indicates a math skill. We showed the structure of this map in Fig. 6.1.

The graph structure contains two major pieces of information about mathematics. One piece is about investigating students' learning progression on series of knowledge with regard to understanding rational numbers, relationship between rational numbers and number line, till further down the path about conducting rational number operations and applying the knowledge about rational numbers to the real world problems. The other piece is about probing the relationship between fractions and rational numbers, and further down the path to be able to use properties of exponents. The two pieces joint at a place where students are expected to understand the reason that sum of two rational numbers is a rational number and then they each develop further independently.

This data set has two parts: generated synthetic binary observations and node descriptions, which are short descriptions in words about nodes.

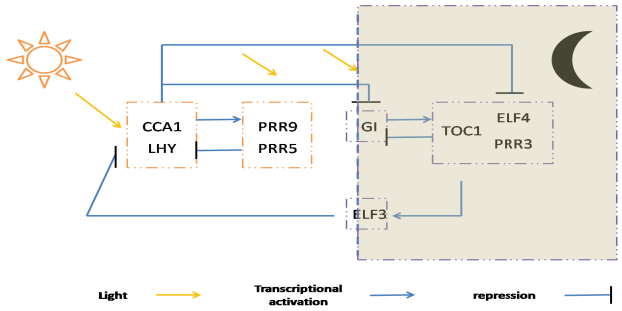


Figure 6.2: The A. thaliana oscillator loops of the circadian clock network.

Arabidopsis Thaliana Circadian Regulation Gene Data. We use the Arabidopsis Thaliana Circadian gene expression data investigated in [102]. The data sets consist of 9 genes, *LHY*, *CCA1*, *TOC1*, *ELF4*, *ELF3*, *GI*, *PRR9*, *PRR5*, and *PRR3*. The group of genes create transcriptional feedback loops and are critical to understand the internal clock-signalling network of plant. The Arabidopsis data are sampled from two light-dark conditions: (I) 10h:10h light/dark cycle and (II) 14h:14h light/dark cycle. Each data set contains 13 time points collected with the interval of 2 hours. We build a gold standard network based on the biological literatures [183, 228, 62, 208, 110, 190]. In this network, CCA1 and LHY proteins directly bind to the promoter of TOC1 to represses the expression of TOC1. The pseudo-response regulators PRR5 and PRR9 are activated by CCA1 and LHY and repress CCA1 and LHY subsequently. G1 improves the expression of TOC1. ELF4 is repressed by CCA1. In Figure 6.2, we show a detailed referred graph figure used in our previous work [141, 145].

We collected the text data for these genes independently from the literatures for retrieving the network. All the descriptions of these 9 genes are from the Arabidopsis Information Resource Database [1], which are short summarized key words related to these genes.

In addition, we revised Figure 6.2 into an acyclic directed graph showed in Figure 6.3. We used this revised network to generate another synthetic data set with 20 observations.

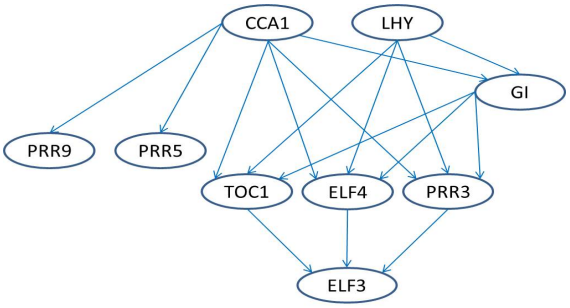


Figure 6.3: The A. thaliana synthetic network.

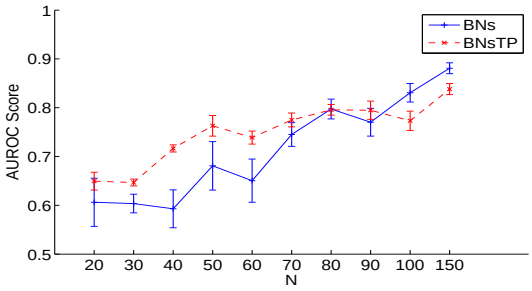


Figure 6.4: The AUROC score over the number of samples.

6.2.2 Experimental Results

We generated a group of education synthetic data with different numbers (N) of observations and tested the structure prediction performance on them. For each data set, we did the experiments 10 times independently. We showed the AUROC score changes over N in Figure 6.4. BNsTP outperformed BNs when N is small ($N = 20 \sim 70$). This finding helps to setup the fitful scenarios for BNsTP to be applications with limited samples. When more observations are available, the text encoded priors will slowly lose their advantages.

6.2.2.1 Structure Prediction Accuracy

We evaluated the network structure prediction accuracy based on AUROC score in two synthetic data sets and two real biological data sets. For the education synthetic data, both BNsTR and BNs ran 400,000 iterations and used 80,000 iterations for burn-in. For the all thaliana synthetic and real data, both BNsTR and BNs sampled 1,000,000 iterations and took 200,000 iterations for burn-in.

Table 6.1: The AUROC values of BNsTR and BNs on synthetic and real data

	Education Synthetic Data (N=40)	Thaliana Synthetic Data (N=20)	Thaliana T20 (N=13)	Thaliana T28 (N=13)
BNsTR	0.7166	0.6829	0.6170	0.5947
BNs	0.5929	0.4376	0.4090	0.4268

TP, true positive; FP, false positive; TN, true negative; FN, false negative.
Sensitivity = TP/(TP+FN).
Specificity = TN/(TN+FP).
Complementary Specificity = 1- Specificity = FP/(TN+FP).
The ROC curves are plotted with the Sensitivity scores against the corresponding Complementary Specificity scores.

We showed the results in Table 6.1. In all three data sets, BNsTR performed much better than the general BNs approach. For example, BNsTR beat BNs with 0.1237 improvement in education synthetic data, 0.2453 improvement in thaliana synthetic data, 0.2080 improvement in thaliana T20 data, and 0.1679 improvement in thaliana T28 data. Both BNsTR and BNs methods get better scores in thaliana synthetic data than real data. Our explanations are that the real microarray gene expression data is usually more noisy than the synthetic observations and the synthetic data have more data points. When considering the difference between the synthetic and real thaliana networks, the acyclic constraint of Bayesian Networks may be the additional reason to affect the prediction performance in real data.

All our experimental results showed that the text information of nodes in both education and biological applications greatly improve the structure prediction in the scenarios of limited samples. In this paper, we had two assumptions. First, we assumed that the professional descriptions of items from domain experts may contain rich structured information, that is, that more similar functional entities would be described with more identical words. Second, we assumed that neighbored nodes in network structures have similar functional roles. Third, when two functions' definition in texts and networks are consistent, the inherent structural information in text will be useful prior to limit the model searching space in network structure inference procedure. Our findings during our experimental study demonstrated these

previous assumptions.

Chapter 7

Online Spectral Clustering on Unlimited Graph Streams

With the fast accumulation of stream data from various type of networks, significant research interests have arisen in applying spectral clustering on evolving graphs. However, the common limitation of existing spectral methods is their computational inefficiency. Most of them are off-line methods and need to recompute solutions from scratch at each time point. The existing work on improving the computational performance of spectral clustering is very limited. The application of the existing incremental spectral clustering approach is limited by its scalability constraint. We propose our computationally efficient online spectral clustering method ISSUER (**I**ncremental **S**pectral clu**S**tering based on matrix pert**U**rbation th**E**o**R**y) with three novel spectrum approximation algorithms: FOA (First Order Approximation), GEPT (General Eigen Perturbation Theory) and EEPT (Enhanced Eigen Perturbation Theory) [142]. Our experimental study shows that our approaches outperform the existing incremental spectral clustering approach in computational efficiency while having better clustering accuracy.

7.1 Related Work

We summarize our related work into two parts: incremental spectral clustering and the related area, evolutionary spectral clustering. For general graph clustering and spectral clustering techniques, we refer to [231, 196] for recent surveys.

7.1.1 Incremental Spectral Clustering

Incremental clustering are usually used to handle two types of clustering tasks [266]: (i) that sequentially clusters incoming new data points that are each observed once, known as data stream clustering [40]; (ii) that continuously clusters data points that are each observed at multiple time points. The incremental clustering tasks are mainly focusing on high computational efficiency.

The existing work on incremental spectral clustering have [248, 158, 203]. Both Valgrem's work [248] and Kong's work [158] targeted the first type of task. They reduced the computational cost by incrementally approximating original large affinity matrices with the smaller ones that only consist of the representative points of clusters. Their methods are designed to handle the insertion and deletion of objects. Ning's work [203] focused on the second type of task. It reduced the computational cost by incrementally updating the eigenvalues/vectors by using similarity change operations on incidence matrix. Their eigenvalue approximation is the first order Taylor approximation. Their perturbed eigenvectors is estimated based on their empirical finding that only the neighborhoods of the nodes connecting the changed edges contribute to the changes of perturbed eigenvectors. The computational gain of Ning's method is only obtained in the condition that a matrix perturbation related to affinity matrices or adjacency matrices of graphs consists of very limited number of similarity changes.

Although the existing work on incremental spectral clustering are very limited, there is a large body of work on data stream incremental clustering other than spectrum analy-

sis, such as, incremental hierarchical clustering [41], incremental micro-clustering [176], and incremental correlation clustering [184] and reference therein.

7.1.2 Evolutionary Spectral Clustering

Evolutionary clustering aims to discover clusters in a sequence of clustering tasks from time series data [37]. It is designed specifically for time series data with slowly drifting clustering boundaries and use temporal smoothness functions to eliminate short-term noises. In recent years, this approach has greatly expanded in classical spectral clustering algorithms, such as [49, 266, 247, 99].

Evolutionary spectral clustering is highly correlated to our research. From the view of the matrix perturbation theory [244], this application setting is seen as small perturbations. With a sufficiently small perturbation, the perturbed eigensystem does not deviate far from the existing one. Hence, the evolutionary scenario provides a good condition for the success of computationally efficient incremental spectrum computation techniques.

7.2 Preliminary and Background

7.2.1 Preliminary

A **weighted undirected graph** G is a 3-tuple $G = (V, E, W)$ where V is the set of vertices of G and $E \subseteq V \times V$ is the set of undirected edges of G with $(u, v) \equiv (v, u) : u, v \in V$. $W : V \times V \rightarrow R^+ \cup \{0\}$ is the function assigning a non-negative real value $W(l)$ to each node pair $l \in V \times V$. If a link $l \in E$, then $W(l) > 0$, otherwise $W(l) = 0$. An **evolving weighted undirected graph** $< G >$ is a sequence of weighted undirected graphs $< G_1, G_2, \ldots, G_T >$, where $V_{G_1} = \cdots = V_{G_T} = V$. For simplicity, in the remaining of the paper we write evolving graph to denote evolving weighted undirected graph.

The **adjacency matrix** $A(G)$ of a weighted undirected graph G with n nodes is an $n \times n$ matrix, where each entry $a_{i,j} = W(i, j)$. A is a symmetric matrix. The **graph Laplacian**

matrix L for a given G is defined as $L(G) = D(G) - A(G)$, where $D(G)$ is a diagonal matrix with $d_{i,i} = \sum_j a_{i,j}$. The **transition matrix** $B(G) = D^{-1} \times A(G)$. Correspondingly we define an **evolving matrix** $< A >$ as $< A_1, A_2, \ldots, A_T >$.

7.2.2 Spectral Clustering

The spectral clustering algorithms are based on spectral graph theory, where the study of eigenvalues/eigenvectors of the affiliated squared matrices such as L or B that represent graphs are the essence.

Without losing the generality, in this paper we focus on the Normalized Cut algorithm [238, 18]. Our algorithm can easily extend to other spectral clustering methods.

In k-way normalized cut, the algorithm minimizes the objective function as $NC = \sum_{i=1}^{k} \frac{assoc(S_i, V \backslash S_i)}{assoc(S_i, V)}$, where $assoc(S_i, S_j) = \sum_{p \in S_i, q \in S_j} W(p, q)$ and $V \backslash S_i$ is the complement of S_i. In Bach's work [18], it showed that the NC measure is equal to $k - trace(X^T D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X)$, where X is a $n \times k$ matrix with two constraints: (i) the columns of $X^T D^{-\frac{1}{2}}$ piecewise constant w.r.t. the set S and (ii) $X^T X = I$. X is a normalized indicator matrix. The solution for the relaxed version of NC optimization problem is the eigenvectors corresponding to k largest eigenvalues of the matrix $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ denoted as A'.

The general procedure of k-way normalized cut algorithm is as follows:

1. compute the eigenvectors X pertaining to the largest k eigenvalues of the matrix $A' = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ or its variants such as the transition matrix B.

2. let X be the matrix with each row as the feature vector for a node (or data point).

3. run the k-means clustering method to cluster the nodes (or data points).

Figure 7.1 depicts an evolving graph with two snapshots. There are clearly two clusters in each graph ($k = 2$). The red dashed lines in Figure 7.1 indicate the partitioning boundary of the clusters.

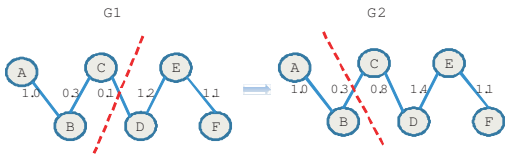


Figure 7.1: An example of an evolving graph with two snapshots

In this paper, we are focusing on the first step of the general algorithm and propose multiple novel methods to incrementally calculate the eigenpairs to save computational time. We formalize our incremental spectral updating problem as follows:

In a smoothly evolving matrix $< A' >$, given A'_t, its eigenvalues/vectors λ_t, x_t, and A'_{t+1}, incrementally compute the corresponding eigenpair: λ_{t+1}, x_{t+1} of A'_{t+1}.

7.3 Methods

In this section, we will discuss our three novel incremental spectral updating algorithms: FOA, GEPT, and EEPT.

7.3.1 First Order Approximation (FOA) Approach

Suppose that λ and x is an eigen-pair of a matrix A' with $x^T x = 1$. Given a new symmetric matrix $\tilde{A}' = A' + \Delta A'$, we denote the new eigenpair as $\tilde{\lambda} = \lambda + \Delta\lambda$ and $\tilde{x} = x + \Delta x$ and have $(A' + \Delta A')(x + \Delta x) = (\lambda + \Delta\lambda)(x + \Delta x)$.

We ignore the second order components $\Delta\lambda\Delta x$ and $\Delta A'\Delta x$ and with $A'x = \lambda x$ obtain

$$A'\Delta x + \Delta A' x = \lambda\Delta x + \Delta\lambda x. \tag{7.1}$$

By multiplying x^T to the left of both sides of the equation, we have

$$\Delta\lambda = x^T \Delta A' x. \tag{7.2}$$

We substitute $\Delta\lambda$ in Equation 7.1 with Equation 7.2 and have

$$(A'-\lambda I)\Delta x=((x^T\Delta A'x)I-\Delta A')x. \tag{7.3}$$

The matrix $A'-\lambda I$ is singular. There is no unique solution for Δx. To address this issue, Ning et al. [203] adopted an approximation approach that only keeps the corresponding rows and columns of the neighbors of the nodes in a similarity change. It created a smaller matrix $K_{N_{ij}}$ to replace $A'-\lambda I$. We propose a new method that substitutes the matrix A' in Equation 7.3 with the new matrix $\tilde{A}'$ and have

$$\Delta x=(\tilde{A}'-\lambda I)^{-1}((x^T\Delta A'x)I-\Delta A')x. \tag{7.4}$$

In Example 1, we show the perturbed eigenpair approximation errors Err_λ and Err_X of graph $G2$ in Figure 7.1 based on FOA. The measures of spectrum approximation errors will be discussed in Section 7.4.2. For the purpose of comparison, we included the eigenpair approximation errors of Ning's approach [203] denoted as SC-FOA (Similarity Change based First Order Approximation approach).

Example 1. *The spectrum approximation errors of the estimated k eigenvectors $\hat{X}(G2)$ using Ning's work are $Err_\lambda = 0.1444$ and $Err_X = 28.36^o$*

The spectrum approximation errors of the estimated k eigenvectors $\hat{X}(G2)$ using FOA are $Err_\lambda = 0.0399$ and $Err_X = 10.45^o$

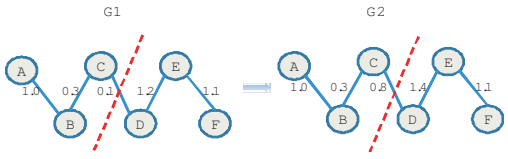


Figure 7.2: The clustering results of the evolving graph in Figure 7.1 by using SC-FOA

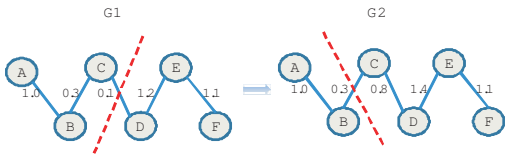


Figure 7.3: The clustering results of the evolving graph in Figure 7.1 by using FOA

Figure 7.2 and 7.3 depict the clustering results of the evolving graph $G1$ and $G2$, respectively using SC-FOA and FOA. SC-FOA does not partition $G2$ correctly while FOA does.

7.3.2 Eigen Perturbation Theory Based Approaches

The problem of how to compute the perturbation of characteristic values and vectors of A' given a small matrix variation $\Delta A'$ is a typical perturbation expansion problem well investigated in the field of Matrix Perturbation Theory. In this subsection, we propose two additional incremental eigenpair updating approaches based on the existing work on eigen perturbation theory [264, 244].

7.3.2.1 General Eigen Perturbation Theory (GEPT) Approach

Our GEPT approach used Gerschgorin's Theorem [89, 264] and Stewart's invariant subspace perturbation theorem [244] to estimate the perturbed eigenvalues and eigenvectors respectively.

Theorem 5. *Gerschgorin's Theorem. Given an $n \times n$ matrix A, let*

$$a_i = \sum_{j \neq i} |a_{i,j}| \quad and \quad Q_i = \{x: \ |x - a_{ii}| \leq a_i\},$$

that is called a Gerschgorin disk. Then the spectrum

$$\mathcal{L}(A') \subset \bigcup_{i=1}^{n} Q_i.$$

Moreover, if m of the Gerschgorin disks are isolated from other $n-m$ disks, there are precisely m eigenvalues of A in the union of the m Gerschgorin disks.$\diamond$

By shrinking a Gerschgorin disk while keeping separating it from other disks, we could get a sharp bound for an eigenvalue. We followed the procedure showed in [264] to derive a perturbed simple eigenvalue $\lambda_1(\tilde{A}')$ as follows.

$$\lambda_1(\tilde{A}')=\lambda_1(A')+x^T\Delta A'x+\mathcal{O}(\|\Delta A'\|^2). \tag{7.5}$$

By ignoring the second order term, we have the approximation solution of a general perturbed eigenvalue $\tilde{\lambda}$ as

$$\tilde{\lambda}\approx\lambda+x^T\Delta A'x. \tag{7.6}$$

The incremental eigenvector updating in our GEPT approach is based on Stewart's work on invariant subspace perturbation theory [244]. Given a subspace χ, when $A'\chi \subset \chi$ where $A'\chi = \{A'\mu : \mu \in \chi\}$, we call χ an invariant subspace of A'. We denote X as the matrix with k eigenvector of A' as columns and have the invariant subspace χ spanned by the columns of X denoted as $\chi(X)$. The columns of X form a basis for χ and there is a unique matrix L such that $A'X = XL$. L is called the representation of A' in χ w.r.t. X and has the same eigenvalues corresponding to X. Here we only consider the invariant subspace with the spectrum of L separated from other eigenvalues of A'. We call such a subspace simple invariant subspace. In practical spectrum calculation, considering round-off errors in numerical methods, we could see our selected k-eigenvalues always simple eigenvalues that are separated from other eigenvalues.

Theorem 6. *Suppose two matrices X and Y. Let the columns of X be linearly independent and the columns of Y span the orthogonal complement of a subspace χ. Then χ is an invariant subspace of A' if and only if $Y^TA'X=0$.*$\diamond$

Theorem 7. *Stewart's Invariant Subspace Perturbation Theorem. Let* $[X\ Y]$ *be orthogonal and suppose that* $\chi(X)$ *is a simple invariant subspace of* A' *spanned by* X*, so that*

$$[X\ Y]^T A'[X\ Y]=\begin{bmatrix} L_1 & 0 \\ 0 & L_2 \end{bmatrix},$$

where $L_1=X^T A'X$ *and* $L_2=Y^T A'Y$*.*

Given a perturbation $\Delta A'$*, let*

$$[X\ Y]^T \Delta A'[X\ Y]=\begin{bmatrix} E_{11} & E_{12} \\ E_{12}^T & E_{22} \end{bmatrix}.$$

Let $\gamma=\eta=\|E_{12}\|_2$ *and* $\sigma=sep(L_1,L_2)-2\|E_{12}\|_2$*, where* $sep(L_1,L_2)=inf_{\|P\|_2=1}\|PL_1-L_2P\|_2>0$*. Then if* $\frac{\gamma}{\sigma}<\frac{1}{2}$*, there is a unique matrix* P *satisfying* $\|P\|_2\leq\frac{2\gamma}{\sigma+\sqrt{\sigma^2-4\gamma^2}}<2\frac{\gamma}{\sigma}$*, such that the columns of*

$$\tilde{X}=(X+YP)(I+P^TP)^{-\frac{1}{2}} \tag{7.7}$$

$$\tilde{Y}=(Y-XP^T)(I+PP^T)^{-\frac{1}{2}} \tag{7.8}$$

form orthonormal bases for simple invariant subspace $\tilde{\chi}$ *of* $\tilde{A}'=A'+\Delta A'$ *and the orthogonal supplement of* $\tilde{\chi}$*.*$\diamond$

We let $X=x$ be an eigenvector corresponding to a simple λ. According to Theorem 6 and 7, we get the approximate solutions for $\tilde{x}$ and $\tilde{Y}$ as follows.

$$\tilde{x}\approx x+Yp=x+Y(\lambda I-L_2)^{-1}Y^T\Delta A'x \tag{7.9}$$

$$\tilde{Y}\approx Y-xp^T=Y-xx^T\Delta A'Y(\lambda I-L_2)^{-1}. \tag{7.10}$$

Due to the limitation of the space, we omitted the mathematical derivation and proofs. More technical details are available on [89, 264, 244].

In Example 2, we show the perturbed eigenpair approximation errors of graph $G2$ in

Figure 7.1 based on GEPT.

Example 2. *The spectrum approximation errors of the estimated k eigenvectors $\hat{X}(G2)$ using GEPT are $Err_\lambda = 0.0399$ and $Err_X = 13.88^o$*

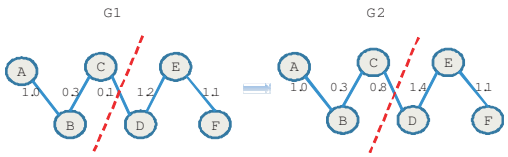


Figure 7.4: The clustering results of the evolving graph in Figure 7.1 by using GEPT

Figure 7.4 depicts the clustering results of the evolving graph $G1$ and $G2$ using GEPT. GEPT does not partition the graph $G2$ correctly.

7.3.2.2 Enhanced Eigen Perturbation Theory (EEPT) Approach

The eigen perturbation theory discussed in the GEPT approach builds a strong theoretic foundation for our incremental eigenpair updating problem. It not only provides us solutions to approximate perturbed eigenvalues/vectors but also shapes them with informative perturbation bounds. However the eigen updating technique used in GEPT has obvious drawbacks. First, in evolving scenarios, for each of k selected eigenvectors x, the GEPT algorithm needs to continuously store and update their corresponding Ys. It may eliminate the computational benefits obtained by our eigen incremental updating scheme with the increasing cluster number. Second, by approximating eigenvalues/vectors separately, it may accumulate errors quickly. Third, the structure of the symmetric matrix A' is not fully explored.

Here we show our third approach EEPT that is based on the same perturbation theory [244] as GEPT. But it is computationally more efficient and has better accuracy on eigen perturbation estimation. Different from GEPT, it solves both eigenvalue and eigenvector updating problem in the same theoretic framework.

Let X be the largest k eigenvectors of A' and Y form a basis of the orthogonal complement of the invariant subspace $\chi(X)$ with $X^TX = I$ and $Y^TY = I$. With Theorem 6, considering that A' is symmetric, we have

$$A'[X\ Y]=[X\ Y]\begin{bmatrix} L_1 & 0 \\ 0 & L_2 \end{bmatrix},$$

where L_1 and L_2 are symmetric. Given a small perturbation $\Delta A'$, we have

$$(A'+\Delta A')[X\ Y]=[X\ Y]\begin{bmatrix} L_1+E_{11} & E_{12} \\ E_{12}^T & L_2+E_{22} \end{bmatrix}. \tag{7.11}$$

By multiplying an orthogonal matrix

$$F=\begin{bmatrix} I & -P^T \\ P & I \end{bmatrix}\begin{bmatrix} (I+P^TP)^{-\frac{1}{2}} & 0 \\ 0 & (I+PP^T)^{-\frac{1}{2}} \end{bmatrix}$$

on both sides of Equation 7.11, we have

$$(A'+\Delta A')[X\ Y]F=[X\ Y]FF^T\begin{bmatrix} L_1+E_{11} & E_{12} \\ E_{12}^T & L_2+E_{22} \end{bmatrix}F. \tag{7.12}$$

With a well selected P, we let

$$F^T\begin{bmatrix} L_1+E_{11} & E_{12} \\ E_{12}^T & L_2+E_{22} \end{bmatrix}F=\begin{bmatrix} \tilde{L_1} & 0 \\ 0 & \tilde{L_2} \end{bmatrix}. \tag{7.13}$$

Hence, Equation 7.12 has the form

$$(A'+\Delta A')[X\ Y]F=[X\ Y]F\begin{bmatrix} \tilde{L_1} & 0 \\ 0 & \tilde{L_2} \end{bmatrix}. \tag{7.14}$$

It satisfies the condition in Theorem 6. Hence, we get Equation 7.7 and 7.8.

In order to satisfy Equation 7.13, we need

$$-P(L_1+E_{11})+E_{12}^T-PE_{12}P+(L_2+E_{22})P=0 \tag{7.15}$$

$$or\ \ P(L_1+E_{11}+E_{12}P)=E_{12}^T+(L_2+E_{22})P, \tag{7.16}$$

which is the direct result by expanding Equation 7.13. By ignoring the higher order term in the Equation 7.15 and reorganize it, we have

$$P(L_1+E_{11})-(L_2+E_{22})P=E_{12}^T. \tag{7.17}$$

Equation 7.17 is Sylvester's Equation that has the form $AX - XB = C$. We show the necessary and sufficient condition to have a unique solution for Sylvester's Equation [244] below.

Theorem 8. *The Sylvester's Equation $AX - XB = C$ has a unique solution if and only if $\mathcal{L}(A) \cap \mathcal{L}(B) = \emptyset$, where $\mathcal{L}(.)$ is the spectrum of a square matrix.*$\diamond$

From the perturbation theory point of view to understand the spectral clustering [253], the selected k eigenvectors X of the matrix A are the perturbed indicator vectors of k clusters $\bar{X}$ of the corresponding matrix $\bar{A}$ with a small perturbation $\Delta\bar{A} = A - \bar{A}$. We used canonical angles Θ between subspaces $\chi(X)$ and $\chi(\bar{X})$ to measure the difference between X and $\bar{X}$. We let $\sin\Theta$ be the diagonal matrix with the sine of Θ as the diagonal entries. The Davis-Kahan Theorem [244] shows that $\|\sin\Theta\| \leq \frac{\|\Delta\bar{A}\|}{\sigma}$, where $\|\cdot\|$ is Frobenius norm and $\sigma = |\lambda_k - \lambda_{k+1}|$, where λ_k and λ_{k+1} are the largest kth and $(k+1)$th eigenvalues. The smaller $\|\sin\Theta\|$ represents the smaller difference between X and $\bar{X}$, and indicates the better clustering output. Both Meila's and Ng's work [237, 200] showed that a larger eigengap $|\lambda_k - \lambda_{k+1}|$ between the selected k eigenvalues $\Lambda^{1\cdots k} = \{\lambda_1, \cdots, \lambda_k\}$ and other $n-k$ eigenvalues $\mathcal{L}(A')\backslash\Lambda^{1\cdots k}$g provides a better clustering results.

L_1 and L_2, as the representation of A' in the invariant subspace w.r.t. X or Y, respectively reserves the spectrums $\Lambda^{1\cdots k}$ and $\mathcal{L}(A')\backslash\Lambda^{1\cdots k}$. With a well selected k eigenvalues with a large gap $|\lambda_k - \lambda_{k+1}|$, it guarantees that $\mathcal{L}(L_1)$ and $\mathcal{L}(L_2)$ are well separated. With a sufficiently small perturbation $\Delta A'$, $\mathcal{L}(\tilde{L_1})$ and $\mathcal{L}(\tilde{L_2})$ are closer enough to $\mathcal{L}(L_1)$ and $\mathcal{L}(L_2)$ so that $\mathcal{L}(\tilde{L_1})$ and $\mathcal{L}(\tilde{L_2})$ are still separated. Hence, there is a unique solution for Equation 7.17. Algorithms to solve Sylvester's Equation are available in [214] and reference therein.

With Equation 7.13 and Equation 7.16, we have

$$\tilde{L_1} \approx L_1 + \Delta E_{11} + E_{12} P, \tag{7.18}$$

$$\tilde{L_2} = L_2 + \tilde{Y}^T \Delta A' \tilde{Y} \approx L_2 + Y^T \Delta A' Y. \tag{7.19}$$

Let $\Lambda^{1\cdots k}(\tilde{L_1})$ and U be the eigenvalues and its corresponding normalized eigenvectors of $\tilde{L_1}$. We have the perturbed k eigenvalues $\tilde{\Lambda}^{1\cdots k}$ and eigenvectors $\tilde{X}$ as

$$\tilde{\Lambda}^{1\cdots k} = \Lambda^{1\cdots k}(\tilde{L_1}) \tag{7.20}$$

$$\tilde{X} = \hat{X} U = (X + YP)(1 + P^T P)^{-\frac{1}{2}} U \approx (X + YP) U. \tag{7.21}$$

In Example 3, we show the perturbed eigenpair approximation errors of graph $G2$ in Figure 7.1 based on EEPT.

Example 3. *The spectrum approximation errors of the estimated k eigenvectors $\hat{X}(G2)$ using EEPT are $Err_\lambda = 0.0023$ and $Err_X = 1.1401^o$*

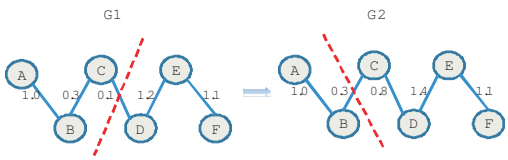


Figure 7.5: The clustering results of the evolving graph in Figure 7.1 by using EEPT

Figure 7.5 depicts the clustering results of the evolving graph $G1$ and $G2$ using EEPT. EEPT correctly partitioned the graph $G2$.

7.3.3 Clustering Re-initialization Policies

Considering the dramatic fall in clustering accuracy caused by accumulated approximation errors or a sudden and dramatic shift of cluster boundary, it is necessary to re-initialize the clustering for all the approximation techniques.

We measure the shifting level of cluster boundaries by using the maximum canonical angle θ between invariant subspaces spanned by k-eigenvectors X_t and X_{t+1} over successive time points. In EEPT, we approximate θ with $\tilde{\theta}$ between estimated $\tilde{X_t}$ and $\tilde{X_{t+1}}$. We use $\tilde{\theta}$ to measure both the accumulated approximation errors and large perturbation.

We listed the re-initialization policies for our proposed approximation technique and SC-FOA [203] as follows.

1. EEPT: re-initialize under the conditions: (i) the maximum canonical angle between the subspaces $\chi(\tilde{X_t})$ and $\chi(\tilde{X}_{t+1})$ is more than a given threshold σ; (ii) $\tilde{L_1}$ has non-real eigenvalues.

2. GEPT: re-initialize under the conditions: (i) after an arbitrary number of time points n_{limit}.

3. FOA: re-initialize under the conditions: (i) after an arbitrary number of time points n_{limit}; (ii) $(\tilde{A}' - \lambda I)$ is singular.

4. SC-FOA: re-initialize under the conditions: (i) after an arbitrary number of similarity change operations n_{limit}; (ii) $K_{N_{ij}}$ is singular.

7.3.4 Time Complexity Analysis

The standard eigenvalue solver takes $\mathcal{O}(N^3)$ operations [237], where N is the number of graph nodes. When A' and its resulting eigen-system are very sparse, Lanczos method [96] reduces the time complexity to $\mathcal{O}(N^{\frac{3}{2}})$. Ning's method [203] has a approximately linear

time complexity $\mathcal{O}(N)$ when N is large enough and A' is very sparse. However, our experimental study shows that with increasing number of similarity changes in a single matrix perturbation, it performs even worse than the general approach using Lanczos algorithm.

Before we show the time complexities of our three spectrum approximation algorithms, we have the following assumptions: A' are large sparse matrices with the non-zero entries linearly to N. $\Delta A'$ are sparse matrices with the number of non-zero entries $m : m \ll N$; x are vectors with the number of non-zeros entries p; X and Y are sparse matrices with the number of non-zero entries kq and p. Such assumptions are usually the case in our application scenarios.

- FOA: $\mathcal{O}(m)$ on $\Delta\lambda$; for Δx, $\mathcal{O}(N)$ on $((x^T \Delta A' x)I - \Delta A')x$, $\mathcal{O}(N^{2.376})$ on the inverse of $(\tilde{A}' - \lambda I)$ [61], $\mathcal{O}(N^2)$ on $((x^T \Delta A' x)I - \Delta A')x$ and $(\tilde{A}' - \lambda I)$.

- GEPT: $\mathcal{O}(m)$ on $\Delta\lambda$; with the naive sparse matrix multiplication algorithm [109], $\mathcal{O}(\min(p, N) \cdot m \cdot \min(p, N-1))$ on $\tilde{x}$ and $\mathcal{O}(q \cdot \min(m \cdot p, N))$ on $\tilde{Y}$.

- EEPT: The key calculations exist in sparse matrix multiplication and the solver of Sylvester's equation in Equation 7.17. With [109], it takes $\mathcal{O}(k^2 \cdot m)$ on E_{11}, $\mathcal{O}((\min(N-k, p))^2 \cdot m)$ on E_{22}, $\mathcal{O}(k \cdot \min(N-k, p) \cdot m)$ on E_{12}. We assume that P, a $(N-k) \times k$ sparse matrix, has f non-zero entries. It takes $\mathcal{O}(k \cdot f)$ on $\tilde{Y}$, $\mathcal{O}(k \cdot p)$ on $\hat{X}$, $\mathcal{O}(k^3 \cdot p)$ on $\tilde{X}$, $\mathcal{O}(k^2 \cdot \min(N-k, p) \cdot m)$ on $\tilde{L_1}$, and $\mathcal{O}((\min(N-k, p+kf))^2 \cdot m)$ on $\tilde{L_2}$. We treat $L_2 + E_{22}$ and $L_1 + E_{11}$ as diagonal matrices. it takes $\mathcal{O}(k \cdot \min(N-k, p) \cdot m)$ to solve Equation 7.17 based on Hessenberg system [97]. The total time complexity of EEPT is the largest scale of those key calculations.

The time complexities of GEPT and EEPT depend on the values of p and f in matrix multiplication. Parallel algorithms on matrix multiplication is a well-studied problem [107]. Hence, a further computational improvement of our algorithms will be achieved by the parallelization of matrix multiplication in distributed computing environment.

7.4 Experimental Study

We performed all the experiments on a desktop machine with an Intel core i7 2.66 GHz CPU and 6 GB memory. We implemented our three methods in Matlab. We implemented Ning's Method [203] SC-FOA in Matlab.

We compared our three methods: FOA, GEPT and EEPT with the Lanczos algorithm [96] denoted as STANDARD, and Ning's SC-FOA approach. Our experimental study is based on three data sets: one synthetic graph stream, one Facebook social network stream, and one Autonomous System graph stream. We evaluated these methods from two aspects: i) eigenvalue/vector approximation error; ii) clustering performance.

7.4.1 Data Sets

Synthetic Data. We followed the work in [49] to create an evolving graph stream with N nodes and T time points. We evenly divided N nodes into 4 groups. The edges inside each group were randomly selected with the probability Z_{in} and the edges between groups were randomly chosen with the probability Z_{out}. We selected the values of Z_{in} and Z_{out} to let the expected degree of nodes be D_{all} and the expected number of between-group edges connected to each node be D_{out}. Between two successive time points, we randomly selected $r\%$ nodes and changed their memberships randomly.

Real Facebook Data. We used the Facebook social network data that was originally collected and investigated in [252]. We created a data set with 531 users and analyzed their interactions from 03/21/2008 to 01/22/2009. We built the undirected Facebook friendship network G_s between these 531 users from the friendship links. We separated The wall-posts weekly to totally 44 time snapshots and built the undirected user activity network G_p^t from their weekly wall posts. The nodes are users and the edges indicate that there are at least one post between users in a week. Considering the significant lower node degrees in G_p^t than G_s [252], we analyzed a new type of snapshot graph called the constrained user activity

network G^t_{cp} with $A(G^t_{cp}) = A(G_s) + A(G^t_p)$.

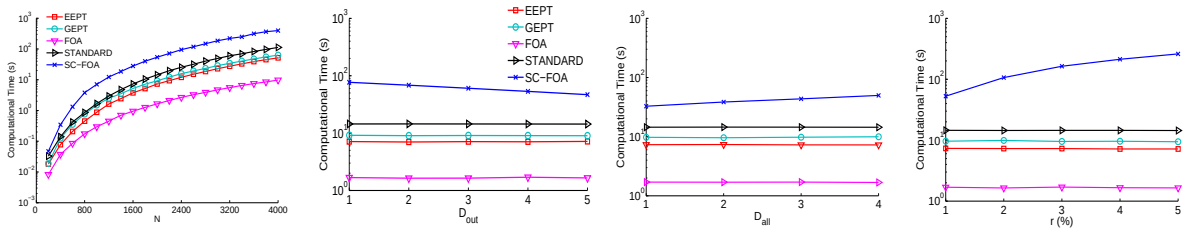
Real Autonomous Systems Data. Border Gateway Protocol (BGP) is an exterior gateway protocol that performs routing between routers of different organizations on Internet that are called Autonomous Systems (ASs) and maintains the routing tables among ASs. An undirected graph is constructed from the BGP routing logs with nodes being ASs and edges indicating the existence of routings between ASs. We used the evolving Autonomous Systems graph data set created by University of Oregon Route Views Project [172].

This Autonomous System graph data contains 733 undirected BGP graphs constructed daily from November 8, 1997 to January 2, 2000. We selected the first 150 graphs from 11/08/1997 to 04/08/1998 and retrieved all edges each occurring in all these graphs. We collected all 2009 nodes pertaining to the shared edges as our vertex set V and analyzed the induced subgraphs over V among these 150 graphs.

7.4.2 Evaluation

Eigenvalue/vector Approximation Errors. We calculated the estimated eigenvalue $\tilde{\lambda}$ error as $Err_\lambda = \max_{\lambda \in \Lambda_k} \mid \tilde{\lambda} - \lambda \mid$. Λ_k are the selected k eigenvalues. Given the true k eigenvectors X, we measure the error Err_X of updated eigenvector $\tilde{X}$ with the maximum canonical angle θ_1 between the subspace $\chi(X)$ and the subspace $\chi(\tilde{X})$ respectively spanned by X and $\tilde{X}$. The nonzero singular values of $Y^T\tilde{X}$ are sines of nonzero canonical angles between $\chi(X)$ and $\chi(\tilde{X})$. Y^T is the orthonormal basis of the orthogonal complement of $\chi(X)$.

Clustering Performance. We evaluated the clustering performance with two measures: clustering accuracy and Rand Index [219]. The clustering accuracy $ACC = N'/N$. With the best match of the target and clustering classes, N' is the number of correctly clustered data, and N is the number of data to be clustered. The Rand index $RI = \frac{TP+TN}{TP+FP+TN+FN}$. TP is the number of pairs of data with the same classes assigned into the same clusters. TN is the number of pairs of data with different classes assigned into different clusters. FP and



(a) Scalability over N (b) Scalability over D_{out} (c) Scalability over D_{all} (d) Scalability over r

Figure 7.6: The computational scalability analysis. (a) The scalability analysis over the graph size N ($D_{all} = 16,\ D_{out} = 4,\ r = 1\%$); (b) Scalability analysis over D_{out} ($N = 2000,\ D_{all} = 16,\ r = 1\%$); (c) Scalability analysis over D_{all} with ($N = 2000,\ D_{out} = 4,\ r = 1\%$); (d) Scalability analysis over r ($N = 2000,\ D_{all} = 16,\ D_{out} = 4$).

FN are computed similarly.

7.4.3 Results

7.4.3.1 Scalability Analysis

We first used the synthetic data to investigate the computational scalability of five methods. We ran all the experiment 10 times independently and calculated the average time.

We created synthetic evolving graphs by varying numbers of nodes N from 200 to 4000 with other parameters in the configuration fixed ($D_{all} = 16$, $D_{out} = 4$, and $r = 1\%$). The scalability experimental results over the graph size N are shown in Figure 7.6a. We found that the SC-FOA approach does not scales well over the graph size N. The computational cost of SC-FOA is proportional to the number of similarity changes n_{sc} and $n_{sc} \approx 2N \cdot r \cdot (D_{all} - D_{out})$. For example, $n_{sc} \approx 48$ with $N = 200$ and $n_{sc} \approx 960$ with $N = 4000$. In all cases, the SC-FOA method performs the worst. With the increasing graph sizes N, our methods keep improving the folds of their computational gain over the standard approach.

By fixing $N = 2000$, $D_{all} = 16$, $r = 1\%$ and varying D_{out} from 1 to 5, we created synthetic data to analyze the scalability over D_{out}. We showed the results in Figure 7.6b. We found that EEPT, GEPT, and FOA are insensitive to the parameter D_{out}. The computational performance of SC-FOA improves with increasing D_{out}. Similar trends were found in the

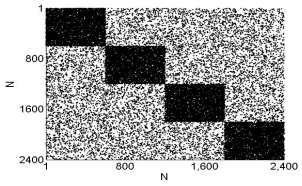


Figure 7.7: The initial adjacency matrix A of a synthetic evolving graph stream with 2400 nodes and 150 time points ($D_{all} = 16,\ D_{out} = 4$), r=1%.

Table 7.1: The comparison of STANDARD, EEPT, GEPT, FOA, SC-FOA on an evolving synthetic network data

	$t_{avg}(s)$	$acc_{avg}(\%)$	$RI_{avg}(\%)$	Err_{λ_avg}	Err_{X_avg} $(0^o - 90^o)$	n_{init}
STANDARD	25.93	98.49	98.91	0	0^o	0
EEPT	12.76	99.16	99.17	0.0392	6.43^o	2
GEPT	16.59	82.13	84.93	0.2418	39.57^o	3
FOA	3.22	53.90	75.40	0.2204	62.17^o	3
SC-FOA	89.23	81.62	84.59	0.2135	42.41^o	3

t_{avg} is the average computational cost for eigenpair calculation over 150 time points . acc_{avg} is the average clustering accuracy. RI_{avg} is the average Rand Index score. n_{init} is the total number of re-initialization.

analysis over D_{all} and r showed in Figure 7.6c and 7.6d. We changed D_{all} from 12 to 16 with the configuration ($N = 2000$, $D_{out} = 4$, $r = 1\%$) and changed r within the range $1\% - 5\%$ with the parameters ($N = 2000$, $D_{all} = 16$, $D_{out} = 4$), we found that EEPT, GEPT, and FOA are flat over D_{all} and SC-FOA takes more time to process graphs with larger D_{all} and r values.

7.4.3.2 Results on a Synthetic Evolving Graph

We created a synthetic evolving graph stream with the configuration $N = 2400$, $T = 150$, $D_{all} = 16$, $D_{out} = 4$, and $r = 1\%$. We showed the adjacency matrix of the initial graph G_0 in Figure 7.7. The piecewise partitioned groups are easily identified from the block diagonal matrix. We evaluated all five methods on the data. The experimental results on spectrum approximation error and clustering performance are shown in Table 7.1. All of our methods run faster than the standard approach while SC-FOA takes more time than STANDARD. We ranked five methods with respect to their computational efficiency in the descending

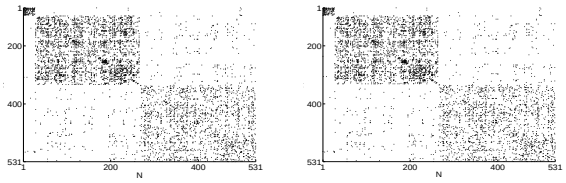
order as: FOA, EEPT, GEPT, STANDARD, SC-FOA. Compared with STANDARD, FOA has as high as more than 8 fold speed-up, EEPT gets more than 2 folds, GEPT gets 1.56 folds, and SC-FOA only has 0.29.

In Table 7.1, we found that larger eigen-pair approximation errors cause worse clustering performance. Such finding indicates that the maximum canonical angle between invariant subspaces may be a useful metric to measure accumulated error. We ranked five methods according to their clustering performance in the descending order as follows: EEPT, STANDARD, GEPT, SC-FOA, FOA. EEPT. EEPT even shows better clustering performance than STANDARD but only uses less than half of its computational time. The reason may be that by incrementally updating the eigenpairs, EEPT incorporates the temporal smoothness into the clustering procedure that adapts well to the evolutionary scenario of our synthetic data. In Ning's work [203], SC-FOA re-initialized after an arbitrary number of similarity changes. In our experiment, for the purpose of having a good picture of the performance of various incremental spectrum approximation techniques over time, we only initialized SC-FOA under the conditions as the follows: (i) $\mid \tilde{\lambda}_t - \lambda_t \mid \geq 1$; (ii) Condition ii in Section 7.3.3. For a fair comparison, we re-initialized our methods GEPT and FOA whenever we restarted SC-FOA. For EEPT, we generally follow the policy described in subsection 7.3.3. In this experiment, EEPT only re-initialized twice at time 72 and 140 while GEPT, FOA and SC-FOA re-initialized 3 times at time 43, 84, and 137.

7.4.3.3 Results on Real Facebook Data

The Facebook friendship network G_s with 531 users consists of 3 communities with 27, 235 and 269 users. We showed the adjacency matrices $A(G_s)$ and $A(G^1_{cp})$ in Figure 7.8. We followed the heuristic showed in [253] to decide the k values. We used $k = 3$ from $t = 1$ to $t = 30$ and $k = 4$ from $t = 31$ to $t = 44$.

With the increasing σ values, the frequency of re-initialization of EEPT decreased, the computational performance improved, and the clustering accuracy dropped. Hence, the



(a) the friendship network(b) a constrained activity network G^1_{cp}

Figure 7.8: The adjacency matrices of Facebook social network and constrained activity network between 03/21/2008-03/28/2008

Table 7.2: The comparison of STANDARD, EEPT ($\sigma = 20$), GEPT, FOA, SC-FOA on real Facebook wallposting data

	t_{avg} (s)	$acc_{avg}(\%)$	$RI_{avg}(\%)$	n_{init}
STANDARD	0.3135	100	100	0
EEPT ($\sigma = 20$)	0.2182	96.25	94.91	8
GEPT ($n_{limit} = 3$)	0.2517	85.49	84.77	3
FOA ($n_{limit} = 3$)	0.0869	64.06	64.26	3
SC-FOA ($n_{limit} = 240$)	1.3488	66.47	66.65	3

selection of σ is based on the trade-off between computational time and accuracy.

We selected EEPT ($\sigma = 20$) to compare with GEPT, FOA, and SC-FOA. The Facebook constrained activity networks evolve smoothly most of time. There are only three large θ value jump ($\theta > 20$) at time 31, 37, and 39. The average number of similarity changes between two consecutive matrices is 183. Hence, we set the parameter n_{limit} for GEPT, FOA, and SC-FOA respectively as 14, 14 and 2400.

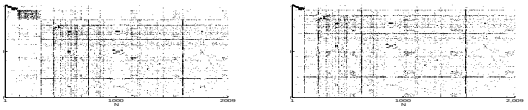
The results in Table 7.2 are consistent to our finding on the synthetic data. Among four incremental approaches, EEPT has the best clustering accuracy and FOA is the most computationally efficient approach. SC-FOA is the most computationally expensive approach.

7.4.3.4 Results on Real ASs Data

Compared with the well-partitioned synthetic graphs and smoothly evolving Facebook data, the Autonomous Systems networks are more noisy. The clustering boundary may shift

Table 7.3: The comparison of STANDARD, EEPT ($\sigma = 25$), GEPT, FOA, SC-FOA on real ASs network data

	t_{avg} (s)	$acc_{avg}(\%)$	$RI_{avg}(\%)$	n_{init}
STANDARD	17.34	100	100	0
EEPT ($\sigma = 25$)	13.34	94.73	92.21	43
GEPT ($n_{limit} = 3$)	18.12	93.33	90.03	37
FOA ($n_{limit} = 3$)	6.52	92.44	88.89	37
SC-FOA ($n_{limit} = 240$)	17.02	93.76	90.67	31



(a) 11/11/1997 (b) 11/12/1997

Figure 7.9: The adjacency matrices of three successive ASs network snapshots collected at time 11/11/1997 and 11/12/1997.

dramatically over time with routing pathes disappearing or emerging swiftly inter-days. In Figure 7.9, we showed an example of dramatic change in the clustering outputs of two successive ASs graph snapshots, where the clusters are the diagonal blocks in matrices.

We chose $k = 5$ through all of the 150 graphs and used EEPT with the configuration $\sigma = 25$ to predict θ. The θ, $\tilde{\theta}$ and σ values are the angle degrees between $0° - 90°$. When $\tilde{\theta}$ is not a real value, we set it as $90°$. We decided the θ and $\tilde{\theta}$ binary levels by using the threshold $\sigma = 25$. We found that in 78% of times $\tilde{\theta}$ correctly predicts the levels of θ.

Same as the Facebook data, we treated the clustering output of STANDARD as the benchmark. We selected EEPT ($\sigma = 25$) to compare with GEPT, FOA, and SC-FOA. We set the parameter n_{limit} for GEPT, FOA, and SC-FOA according to the characteristic of the data. The average time interval between two successive large θ value jump ($\sigma > 25$) is 3.75 and the average number of similarity changes between two consecutive matrices is 73. Hence, we chose GEPT ($n_{limit} = 4$), FOA ($n_{limit} = 4$), and SC-FOA ($n_{limit} = 280$). We showed the results both on computational time and clustering performance in Table 7.3. The results further demonstrated the same finding both in synthetic and real Facebook data. Although the numbers of similarity changes are very small, only 73 between time points, the

computational gain obtained by SC-FOA over STANDARD is still very limited. Hence, it showed that the incremental spectral approach purely based on similarity change insertion or deletion may not work well in many real applications.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In my dissertation, I presented a new online spectral clustering approach on social network streams and our previous work on approximate graph mining with evolutionary process from protein structure database, non-stationary Bayesian Network structure learning from non-stationary microarray gene expression time series data, and Bayesian Network structure learning with text priors imposed by non-parametric hierarchical topic modeling.

In community detection problem of Online Social Networks (OSNs), such as Facebook, Twitter, and Linkedin, spectral clustering techniques have been demonstrated to be very efficient models to identify communities with dense communications inside networks. To address great computational challenges posed by the gigantic number (millions) of users and their interactions in large network streams, we proposed an online spectral clustering method ISSUER with three novel spectrum approximation algorithms: FOA (First Order Approximation), GEPT (General Eigen Perturbation Theory) and EEPT (Enhanced Eigen Perturbation Theory).

In protein motif detection problem, we encoded a protein structure as a geometric graph where a node represents an amino acid residue and an edge represents a physical or a chem-

ical interaction between a pair of residues and encoded structural motifs as subgraphs of a geometric graph. We identified conserved structure fingerprints by searching for frequently occurring approximately subgraphs in a group of graph represented proteins. We devised a frequent subgraph mining algorithm that are capable of identifying salient patterns in large graph database that are otherwise overlooked by using exact matching due to the presence of noises and distortions in the graph databases.

In the non-stationary Bayesian Network learning problem, the microarray gene expression data is characterized by small sample sizes and limited expressed levels. non-stationary Dynamic Bayesian Network (non-stationary DBN) methods are widely used to model the time-varying regulatory networks from non-stationary multivariate microarray time series data. Change-point modeling is a very promising way of dealing with the non-stationarity property. We proposed two new non-stationary algorithms with different change-point detection techniques, Reversible Jump Markov Chain Monte Carlo (RJMCMC) and Perfect Simulation model, to capture the structural dynamics of networks in various biological systems. In addition, we used non-parametric models to encode the unstructured text data to enforce the prior domain knowledge in bayesian network structure learning procedure.

8.2 Future Work

Our future work has several directions. The first direction is parallel algorithm on online spectral clustering methods. In our dissertation, we only show a serial algorithm that consists of key operations that are well studied parallel computing problems. The MapReduce is a parallel programming paradigm proposed by Google [68]. Recently it has been demonstrated to be a very useful technique to improve the computational performance of machine learning and data mining in various parallel platforms, such as clusters [68], GPGPU [114], multi-core [221, 276], and FPGA [235]. Hence, to parallelize our method ISSUER to the MapReduce framework will be further beneficial to massive data processing.

The second direction is bayesian network structure learning on large scale networks inference. The bayesian network structure learning problem is NP-hard [279]. However, most of the existing work could only address networks of small sizes. Recently parallel algorithms on bayesian network learning have started to attract interests of researchers, for example, the parallel dynamic programming algorithm [246] and parallel shortest-path algorithm [180]. Hence, to design an efficient parallel non-stationary bayesian network learning algorithm with bounded error and anytime properties [180] is beneficial in many applications.

The third direction is to use unstructured text information to bias bayesian network structure inference. In many applications, the ground truthes may not be available and may need to be provided by experts based on their latest domain knowledge, such as, biology [227, 102] and education [144]. And experts' domain knowledge evolves over time. Documents, as natural way to encode domain knowledge, could contain rich structured information [26, 25]. Utilizing the text data knowledge with the-state-of-art knowledge provided by experts to bias the structure learning procedure will be beneficial to many applications. Our future work in this field are to build quantitative criteria to judge the quality of the text encoded knowledge.

Bibliography

[1] The arabidopsis information resource (tair) database. `http://www.arabidopsis.org/`.

[2] Dynamic learning maps. `http://www.dynamiclearningmaps.org/`.

[3] Zeinab Abbassi and Vahab S. Mirrokni. A recommender system based on local random walks and spectral methods. In *Proceeding WebKDD/SNA-KDD '07 Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, 2007.

[4] Amira Abdelwahab, Hiroo Sekiya, Ikuo Matsuba, Yasuo Horiuchi, and Shingo Kuroiwa. Collaborative filtering based on an iterative prediction method to alleviate the sparsity problem. In *Proceeding iiWAS '09 Proceedings of the 11th International Conference on Information Integration and Web-based Applications and Services*, 2009.

[5] C. Aggarwal. Social network data analytics. *Springer*, 2011.

[6] C. Aggarwal and H.Wang. Managing and mining graph data. *Springer*, 2010.

[7] C. Aggarwal, N. Ta, J. Feng, J. Wang, and M. J. Zaki. Xproj: A framework for projected structural clustering of xml documents. In *Proceedings of the 2007 ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2007.

[8] C. Aggarwal, N. Ta, J. Feng, J. Wang, and M. J. Zaki. Xproj: A framework for

projected structural clustering of xml documents. In *Proceedings of KDD Conference Proceedings (2007)*, page 46C55, 2007.

[9] C. Aggarwal, Y. Xie, and P. Yu. Gconnect: A connectivity index for massive disk-resident graphs. In *Proceedings of the 2009 Very Large Database (VLDB) Conference*, 2009.

[10] C. Aggarwal, Y. Zhao, and P. Yu. Outlier detection in graph streams. In *Proceedings of the 2011 ICDE Conference*.

[11] C. Aggarwal, Y. Zhao, and P. Yu. On clustering graph streams. In *Proceedings of SIAM Conf. on Data Mining*, page 478C489, 2010.

[12] Charu C. Aggarwal. Managing and mining uncertain data. 2009.

[13] Charu C. Aggarwal, Yan Li, Jianyong Wang, and Jing Wang. Frequent pattern mining with uncertain data. In *Proceedings of the 2009 ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD'09)*, pages 29–37, 2009.

[14] C. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. Koutsoukos. Local causal and markov blanket induction for causal discovery and feature selection for classification part ii: Analysis and extensions. *JMLR*, 11:235C284, 2010.

[15] C. Alpert, A.B. Kahng, and S.Z. Yao. Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, 90:3C26, 1999.

[16] Michelle N. Arbeitman, Eileen E. M. Furlong, Farhad Imam, Eric Johnson, Brian H. Null, Bruce S. Baker, Mark A. Krasnow, Matthew P. Scott, Ronald W. Davis, and Kevin P. White. Gene expression during the life cycle of drosophila melanogaster. *Science*, 297(5590):2270–2275, 2002.

[17] Stephen Aylward and Stephen Pizer. Continuous gaussian mixture modeling. In *Gindi*

(Eds.), Information Processing in Medical Imaging. Springer Lecture Notes in Computer Science 1230, pages 176–189, 1997.

[18] Francis R. Bach and Michael I. Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7, 2006.

[19] D. Bandyopadhyay and J. Snoeyink. Almost-Delaunay simplices : Nearest neighbor relations for imprecise points. In *ACM-SIAM Symposium On Distributed Algorithms*, pages 403–412, 2004.

[20] A.L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 256:509–512, 1999.

[21] Matthew J. Beal, Zoubin Ghahramani, and Carl Edward Rasmussen. Factorial hidden markov models. In *Machine Learning*, pages 29–245. MIT Press, 1997.

[22] Chris A. Benedict, Theresa A. Banks, Lionel Senderowicz, Mira Ko, William J. Britt, Ana Angulo, Peter Ghazal, and Carl F. Ware. Lymphotoxins and cytomegalovirus cooperatively induce interferon-b establishing host-virus detente. *Immunity*, 15:617–626, 2001.

[23] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, 2002.

[24] Allister Bernard and Alexander J. Hartemink. Informative structure priors: joint learning of dynamic regulatory networks from multiple types of data. *Proceedings of Pacific Symposium on Biocomputing*, pages 459–70, 2005.

[25] D. Blei, T. Griffiths, and M. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, 2010.

[26] D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and

the nested chinese restaurant process. In *Procceeding of 2003 Neural Information Processing Systems Conference*, 2003.

[27] C. Borgelt and M. R. Berhold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proceedings of 2nd IEEE International Conference on Data Mining (ICDM 02)*, pages 51–58, 2002.

[28] R. R. Bouckaert. Probabilistic network construction using the minimum description length principle. *Technical report RUU-CS-94-27*.

[29] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163C177, 2001.

[30] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithm for collaborative filtering. In *Proceedings of the 14 th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[31] Derek C. Briggs. Making progress in the modeling of learning progressions. *Learning Progressions in Science*, 4:345–355, 2012.

[32] F. Bromberg, D. Margaritis, and H. V. Efficient markov network structure discovery using independence tests. *JAIR*, 35:449C485, 2009.

[33] Amit Bubna, Sanjiv Ranjan Das, and Nagpurnanand R. Prabhala. Venture capital communities. *Available at SSRN: http://ssrn.com/abstract=1787412 or http://dx.doi.org/10.2139/ssrn.1787412*, 2011.

[34] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

[35] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of 2001 ICDE Conference*, pages 443–452, 2001.

[36] J. Burge and T. Lane. Shrinkage estimator for bayesian network parameters. In *Proceedings of the Eighteenth European Conference on Machine Learning (EMCL 2007)*, 2007.

[37] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In *Proceedings of the 2006 annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[38] Jeffrey Chan, James Bailey, and Christopher Leckie. Discovering correlated spatio-temporal changes in evolving graphs. *Knowledge and Information Systems*, 16(1):53–96, 2008.

[39] P.K. Chan, M.D.F. Schlag, and J.Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, page 1088C1096, 1994.

[40] Moses Charikar, Chandra Chekuri, Tomas Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997.

[41] Moses Charikar, Chandra Chekuri, Tomas Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6), 2004.

[42] C. Chen, C. Lin, M. Fredrikson, M. Christodorescu, X. Yan, and J. Han. Mining graph patterns efficiently via randomized summaries. In *Proceedings of the 2009 Very Large Database (VLDB) Conference*, 2009.

[43] Chen Chen, Xifeng Yan, Feida Zhu, and Jiawei Han. gapprox: Mining frequent approximate patterns from a massive network. In *Proceedings of the 2007 International Conference on Data Mining (ICDM'07)*, 2007.

[44] Zheng Chen and Heng Ji. Graph-based clustering for computational linguistics: A survey. In *Proceedings of TextGraphs-5 - 2010 Workshop on Graph-based Methods for Natural Language Processing*, pages 1–9, 2010.

[45] C.K. Cheng and Y.C.A. Wei. An improved two-way partitioning algorithm with stable performance. *IEEE Transactions Computer-Aided Design*, 10:1502–1511, 1991.

[46] D. Cheng, S. Vempala, R. Kannan, and G. Wang. A divideandmerge methodology for clustering. In *Proceedings of the Twentyfourth Symposium on Principles of Database Systems, ACM Press*, 2005.

[47] J. Cheng and M. J. Druzdzel. Ais-bn: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188, 2000.

[48] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007)*, pages 153–162, 2007.

[49] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L. Tseng. On evolutionary spectral clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(4), November,2009.

[50] S. Chib and E. Greenberg. Understanding the metropolis hasting algorithm. *Amer. Statist*, 49:327–335, 1995.

[51] D. M. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks: search methods and experimental results. In *Learning from Data: Artificial Intelligence and Statistics*, 1996.

[52] D. M. Chickering, D. Heckerman, and C. Meek. A bayesian approach to learning bayesian networks with local structure. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, 1997.

[53] David Maxwell Chickering. Learning equivalence classes of bayesian network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

[54] N. Chopin. Dynamic detection of change points in long time series. *The Annals of the Institute of Statistical Mathematics*, 2006.

[55] F. Chung and L. Lu. Connected components in random graphs with given degree sequences. *Annals of Combinatories*, 6:125–145, 2002.

[56] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review*, 70(6), 2004.

[57] A. Condon and R.M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116C140, 2001.

[58] D. J. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15, 2000.

[59] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[60] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309C347, 1992.

[61] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251C280, 1990.

[62] Michael F. Covington, Satchidananda Panda, Xing Liang Liu, Carl A. Strayer, D. Ry Wagner, and Steve A. Kay. Elf3 modulates resetting of the circadian clock in arabidopsis. *The Plant Cell*, 13:1305–1315, 2001.

[63] Richard M. Cripps, Brian L. Black, Bin Zhao, Ching-Ling Lien, Robert A. Schulz, , and Eric N. Olson. The myogenic regulatory gene mef2 is a direct target for transcriptional activation by twist during drosophila myogenesis. *Genes Dev.*, 12(3):422–34, 1998 Feb. 1.

[64] T. Dalamagas, T. Cheng, K. Winkel, and T. Sellis. Clustering xml documents using structural summaries. *Information Systems, Elsevier*, 2005.

[65] R. Daly, Q. Shen, and S. Aitken. Learning bayesian networks: Approaches and issues. *The Knowledge Engineering Review*, 2009.

[66] J. Davis and P. Domingos. Bottom-up learning of markov network structure. In *Proceedings of the 23rd international conference on Machine learning*, page 271C278, 2010.

[67] L. M. de Campos and J. F. Huete. Approximating causal orderings for bayesian networks using genetic algorithms and simulated annealing. In *Proceedings of the Eight Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 333–340, 2000.

[68] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Sixth Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[69] I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. 7th Int. Conf. on Knowledge Discovery and Data mining*, 2001.

[70] Frank Dondelinger, Sophie Lebre, and Dirk Husmeier. Heterogeneous continuous dynamic bayesian networks with flexible structure and inter-time segment information sharing. In *Proceedings of 2010 International Conference on Machine Learning (ICML10)*, 2010.

[71] L. Donetti and M.A. Munoz. Detecting network communities: A new systematic and efficient algorithm. *Journal of Statistical Mechanics*, 2004.

[72] M. J. Druzdzel. Some properties of joint probability distributions. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 187–194, 1994.

[73] H. Du, M.W. Feldman, S. Li, and X. Jin. An algorithm for detecting community structure of social networks based on prior knowledge and modularity. *Complexity*, 12(3):53–60, 2007.

[74] Hong Duan and Hanh T. Nguyen. Distinct posttranscriptional mechanisms regulate the activity of the zn finger transcription factor lame duck during drosophila myogenesis. *Molecular and Cellular Biology*, 26(4):1414–1423, 2006 Feb.

[75] D. Eaton and K. Murphy. Bayesian structure learning using dynamic programming and mcmc. In *Proceedings of the Twenty-third Annual Conference on Uncertainty in Artificial Intelligence (UAI-07)*, page 101C108, 2007.

[76] Sean R Eddy. Where did the blosum62 alignment score matrix come from. *Nature Biotechnology*, 22:1035 – 1036, 2004.

[77] Paul Fearnhead. Exact and efficient bayesian inference for multiple changepoint problems. *Statistics and Computing*, 16:203–213, 2006.

[78] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 1090C1118, 2002.

[79] Fulvia Ferrazzi, S. Rinaldi, A. Parikh, G. Shaulsky, Blaz Zupan, and Riccardo Bellazzi. Population models to learn bayesian networks from multiple gene expression experiments. *http://www.labmedinfo.org/biblio/author/326*, 2008.

[80] G.W. Flake, R.E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(1):385C408, 2004.

[81] S. Fortunato. Community detection in graphs. *Physics Reports*, page 75C174, 2010.

[82] S. Fortunato, V. Latora, and M. Marchiori. Method to find community structures based on information centrality. *Physical Review*, 2004.

[83] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[84] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[85] W. Fu. Penalized regressions: the bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 1998.

[86] Shouguo Gao and Xujing Wang. Quantitative utilization of prior biological knowledge in the bayesian network modeling of gene expression data. *BMC Bioinformatics*, 2011.

[87] D. Geiger and D. Heckerman. Learning gaussian networks. *Technical report MSR-TR-94-10*, 1994.

[88] Andrew Gelman and Donald B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7:457C472, 1992.

[89] S. A. Gerschgorin. Uber die abgrenzung der eigenwerte einer matrix. *Izv. Akad. Nauk SSSR, Ser. Fiz.-Mat.*, 6:749–754, 1931.

[90] Zoubin Ghahramani. Advanced lectures on machine learning. *Journal of the Royal Statistical Society*, 2004.

[91] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. 2002.

[92] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. In *Proceedings of the National Academy of Sciences*, page 8271C8276, 2001.

[93] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. In *Proceedings of the National Academy of Sciences*, page 8271C8276, 2002.

[94] Christos Gkantsidis, Milena Mihail, and Ellen Zegura. Spectral analysis of internet topologies. In *Proceedings of IEEE INFOCOM 2003*, pages 364 – 374, 2003.

[95] B. Goethals. Frequent set mining. *The Data Mining and Knowledge Discovery Handbook*, Chapter 17:377C397, 2005.

[96] G.H. Golub and C.F. Van Loan. Matrix computations. *John Hopkins Press*, 1989.

[97] G.H. Golub, S. S. Nash, and C. Van Loan. A hessenberg-schur method for the problem ax + xb= c. *IEEE Transactions on Automatic Control*, 24(6):909 – 913, 1979.

[98] Aurelie Goulon, Arthur Duprat, and Gerard Dreyfus. Graph machines and their applications to computer-aided drug design: a new approach to learning from structured data. *Unconventional Computation*, 4135:1–19, 2006.

[99] Nathan Green, Manjeet Rege, Xumin Liu, and Reynold Bailey. Evolutionary spectral co-clustering. In *Proceedings of The 2011 International Joint Conference on Neural Networks (IJCNN)*, 2011.

[100] Peter J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82:711–732, 1995.

[101] Peter J. Green. Trans-dimensional markov chain monte carlo. *Highly Structured Stochastic Systems, Oxford University Press*, 2003.

[102] Marco Grzegorczy, Dirk Husmeier, Kieron D. Edwards, Peter Ghazal, , and Andrew J. Millar. Modelling non-stationary gene regulatory processes with a non-homogeneous bayesian network and the allocation sampler. *Bioinformatics*, 24:2071 – 2078, 2008.

[103] Marco Grzegorczyk and Dirk Husmeier. Improvements in the reconstruction of time-varying gene regulatory networks: dynamic programming and regularization by information sharing among genes. *Journal of Bioinformatics*, 27(5):693–699, 2011.

[104] M. Gu, H. Zha, C. Ding, X. He, and H. Simon. Spectral relaxation models and structure analysis for k-way graph clustering and bi-clustering. *Technical Report*, 2001.

[105] Fan Guo, Steve Hanneke, Wenjie Pu, , and Eric P. Xing. Recovering temporally rewiring networks: A model-based approach. *ICML*, 24, 2007.

[106] Haipeng Guo and William Hsu. A survey of algorithms for real-time bayesian network inference. In *The joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*, 2002.

[107] Anshul Gupta and Vipin Kumar. Scalability of parallel algorithms for matrix multiplication. *International Conference on Parallel Processing*, 3:115 – 123, 1993.

[108] Manish Gupta, Charu C, Aggarwal, Jiawei Han, and Yizhou Sun. Evolutionary clustering and analysis of bibliographic networks. In *Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2011)*, pages 63–70, 2011.

[109] Fred G. Gustavson. Two fast algorithms for sparse matrices: Multiplication and permuted transposition. *ACM Transactions on Mathematical Software (TOMS)*, 4(3), 1978.

[110] Anthony Hall, Laszlo Kozma-Bognar, RekaToth, Ferenc Nagy, and Andrew J. Millar. Conditional circadian regulation of phytochrome a gene expression. *Plant Physiol.*, 127(4):1808–18, 2001.

[111] K.M. Hall. An v-dimensional quadratic placement algorithm. *Management Science*, 17:219–229, 1970.

[112] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery*, 14, 2007.

[113] Alexander J. Hartemink, David K. Gifford, Tommi S. Jaakkola, and Richard A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Proceedings of Pacific Symposium on Biocomputing*, pages 422–433, 2001.

[114] Bingsheng He, Wenbin Fang, Qiong Luo, Naga K. Govindaraju, and Tuyong Wang. Mars: A mapreduce framework on graphics processors. *PACT*, 2008.

[115] D. Heckerman and D. Geiger. Likelihoods and parameter priors for bayesian networks. *Technical report MSR-TR-95-54*, 1995.

[116] David Heckerman. A tutorial on learning with bayesian networks. Technical report, Learning in Graphical Models, 1995.

[117] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[118] R. Hofmann and V. Tresp. Discovering structure in continuous variables using bayesian networks. In *Advances in Neural Information Processing Systems 8 (NIPS'95)*, 1995.

[119] L. B. Holder, D. J. Cook, and S. Djoko. Substructures discovery in the subdue system. *Proc. AAAI'94 Workshop Knowledge Discovery in Databases*, pages 169–180, 1994.

[120] Kenya Honda, Akinori Takaoka, and Tadatsugu Taniguchi. Type i interferon gene induction by the interferon regulatory factor family of transcription factors. *Immunity*, 25:349–360, 2006.

[121] Hsun-Ping Hsieh and Cheng-Te Li. Mining temporal subgraph patterns in heterogeneous information networks. In *Proceedings of IEEE International Conference on*

Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust.

[122] H. Hu, X. Yan, Y. Huang, J. Han, and X.J. Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. In *Proceedings of the 2005 International Conference on Intelligent Systems for Molecular Biology (ISMB'05)*, 2005.

[123] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. In *Proceedings of the 2003 IEEE International Conference on Data Mining (ICDM'03)*, pages 549–552, 2003.

[124] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Proceedings of the 2003 International Conference on Data Mining (ICDM'03)*, 2003.

[125] Jun Huan, Deepak Bandyopadhyay, Jack Snoeyink, Jan Prins, Alex Tropsha, and Wei Wang. Distance-based identification of spatial motifs in proteins using constrained frequent subgraph mining. In *Proceedings of the IEEE Computational Systems Bioinformatics (CSB)*, 2006.

[126] Jun Huan, Jan Prins, Wei Wang, Charlie Carter, and Nikolay V. Dokholyan. Coordinated evolution of protein sequences and structures with structure entropy. In *Computer Science Department Technical Report*, 2006.

[127] Jun Huan, Wei Wang, Deepak Bandyopadhyay, Jack Snoeyink, Jan Prins, and Alexander Tropsha. Mining family specific residue packing patterns from protein structure graphs. In *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 308–315, 2004.

[128] Jun Huan, Wei Wang, Jan Prins, , and Jiong Yang. Spin: Mining maximal frequent

subgraphs from graph databases. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 581–586, 2004.

[129] Dirk Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. *Bioinformatics*, 19:2271–2282, 2003.

[130] Dirk Husmeier, Frank Dondelinger, and Sophie Lebre. Inter-time segment information sharing for non-homogeneous dynamic bayesian networks. In *Proceedings of 2010 Advances in Neural Information Processing Systems (NIPS)*, 2010.

[131] A. Hyvarinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6, 2005.

[132] Katja Ickstadt, Bjorn Bornkamp, Marco Grzegorczyk, Jakob Wieczorek, M. Rahuman Sheriff, Hernan E. Grecco, and Eli Zamir. Nonparametric bayesian networks. *In: Bernardo, Bayarri, Berger, Dawid, Heckerman, Smith, and West (eds): Bayesian Statistics 9, Oxford University Press*, 2010.

[133] Seiya Imoto, Takao Goto, and Satoru Miyano. Estimation of genetic networks and functional structures between genes by using bayesian networks and nonparametric regression. *Proceedings of Pacific Symposium on Biocomputing*, pages 175–186, 2002.

[134] Seiya Imoto, Tomoyuki Higuchi, Takao Goto, Kousuke Tashiro, Satoru Kuhara, and Satoru Miyano. Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. *Computer Society Bioinformatics Conference (CSB'03)*, page 104, 2003.

[135] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceeding of 2000 Practice of Knowledge Discovery in Databases Conference (PKDD'00)*, pages 13–23, 2000.

[136] A.K JAIN, M.N. MURTY, and P.J. FLYNN. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[137] G. M. James, T. J. Hastie, and C. A. Sugar. Principal component models for sparse functional data. *Biometrika*, 87(3):587–602, 2000.

[138] J. N. R. Jeffers. Two case studies in the application of principal component analysis. *Journal of the Royal Statistical Society: Series C*, 16(3):225–236, 1967.

[139] Yi Jia, Vincent Buhr, Jintao Zhang, Jun Huan, and Leonidas N. Carayannopoulos. Comprehensive structural motif mining for better fold annotation. In *Proceedings of the APBC2009 Conference (Asia Pacific Bioinformatics Conference)*, 2009.

[140] Yi Jia, Vincent Buhr, Jintao Zhang, Jun Huan, and Leonidas N. Carayannopoulos. Towards comprehensive structural motif mining for better fold annotation in the "twilight zone" of sequence dissimilarity. *Journal of BMC Bioinformatics*, 10(Suppl 1), 2009.

[141] Yi Jia and Jun Huan. Constructing non-stationary dynamic bayesian networks with a flexible lag choosing mechanism. *BMC Bioinformatics*, 2010.

[142] Yi Jia, Jun Huan, and Hongguo Xu. Issuer: an online spectral clustering method for network data streams. In *Proceedings of the 2012 ICDM Conference (submitted)*, 2012.

[143] Yi Jia, Jun Huan, and Jintao Zhang. An efficient graph mining method for complicated and noisy data with real-world applications. *Knowledge and Information Systems: An International Journal (KAIS)*, 28(2), 2010.

[144] Yi Jia, Tom Walsh, and Fei Zhao. Bayesian network structure learning with text regularity. *Technical Report*, 2012.

[145] Yi Jia, Wenrong Zen, and Jun Huan. Non-stationary bayesian networks based on perfect simulation. In *Proceedings of the 2012 ACM CIKM Conference*, 2012.

[146] M. I. Jordan. Graphical models. *Statistical Science*, 19:140C155, 2004.

[147] M. I. Jordan, Z. Ghahramani, T. S. Jaakola, , and L. K. Saul. An introduction to variational methods for graphical models. *Learning in Graphical Models*, pages 105–161, 1999.

[148] JE Darnell Jr, IM Kerr, and GR Stark. Jak-stat pathways and transcriptional activation in response to ifns and other extracellular signaling proteins. *Science*, 264:1415–1421, 1994.

[149] Kara A. Judson, John M. Lubinski, Ming Jiang, Yueh Chang, Roselyn J. Eisenberg, Gary H. Cohen, and Harvey M. Friedman. Blocking immune evasion as a novel approach for prevention and treatment of herpes simplex virus infection. *J. Virol*, 77:12639–12645, 2003.

[150] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. In *IEEE Symposium on Foundations of Computer Science*, pages 367–377, 2000.

[151] Brian Karrer, Elizaveta Levina, and M. E. J. Newman. Robustness of community structure in networks. *Physical Review*, 77(4), 2008.

[152] Karypis and V. Kumar. Multilevel algorithms for multiconstraint graph partitioning. In *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, pages 343–348, 1999.

[153] Yoshinobu Kawahara and Masashi Sugiyama. Change-point detection in time-series data by direct densityratio estimation. In *Proceedings of 2009 SIAM International Conference on Data Mining (SDM09)*, 2009.

[154] Min-Soo Kim and Jiawei Han. A particle-and-density based evolutionary clustering

method for dynamic networks. In *Proceedings of the 2009 International Conference on Very Large Databases (VLDB)*, volume 2(1), 2009.

[155] Sun Yong Kim, Seiya Imoto, and Satoru Miyano. Inferring gene networks from time series microarray data using dynamic bayesian networks. *Brief Bioinform*, 4:228–235, 2003.

[156] Alexandre Klementiev, Dan Roth, , and Kevin Small. An unsupervised learning algorithm for rank aggregation. In *Proceedings of the 2007 ECML Conference*, 2007.

[157] D. Koller and N. Friedman. Probabilistic graphical models. *Massachusetts: MIT Press*, 2009.

[158] Tengteng Kong, Ye Tian, and Hong Shen. A fast incremental spectral clustering for large data sets. In *Proceedings of International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, 2011.

[159] Mirko Krivanek and Jaroslav Moravek. Np-hard problems in hierarchical-tree clustering. *Journal Acta Informatica*, 23(3), 1986.

[160] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of the 2001 International Conference on Data Mining (ICDM'01)*, pages 313–320, 2001.

[161] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. In *Proceedings of the SIAM International Conference on Data Mining*, 2004.

[162] Mikls Kurucz, Andrs A Benczr, Kroly Csalogny, and Lszl Lukcs. Spectral clustering in social networks. *Advances in Web Mining and Web Usage Analysis*, 91:1–20, 2007.

[163] Mayank Lahiri and Tanya Y. Berger-Wolf. Periodic subgraph mining in dynamic networks. *Knowledge and Information Systems*, Onine First, 09/2009.

[164] Mayank Lahiri and Tanya Y. Berger-Wolf. Structure prediction in temporal networks using frequent subgraphs. *Computational Intelligence and Data Mining, 2007*, pages 35–42, 2007.

[165] K. Lakshmi and T. Meyyappan. Frequent subgraph mining algorithms - a survey and framework for classification. In *Proceedings of First International Conference on Information Technology Convergence and Services (ITCS 2012)*, 2012.

[166] Helge Langsetha, Thomas D. Nielsenb, Rafael Rumi, and Antonio Salmeron. Mixtures of truncated basis functions. *International Journal of Approximate Reasoning*, 53(2):212–227, 2012.

[167] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, and Y. Yurramendi. Learning bayesian network structures by searching for the best ordering with genetic algorithms. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 26(4), pages 487–93, 1996.

[168] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. In *Proceedings of the Royal Statistical Society*, volume 50, pages 154–227, 1988.

[169] M. Lavielle and G. Teyssiree. Adaptive detection of multiple change-points in asset price volatility. *Long Memory in Economics*, page 129C156, 2005.

[170] Sophie Lebre, Jennifer Becq, Frederic Devaux, Michael Stumpf, and Gaelle Lelandais. Statistical inference of the time-varying structure of gene regulation networks. *BMC Systems Biology*, 4(1):130, 2010.

[171] Seak Fei Lei and Jun Huan. Towards site-based function annotations for protein structures. In *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM'08)*, pages 193–198, 2008.

[172] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the 2005 annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2005.

[173] Jure Leskovec, Lada Adamic, and Bernardo Huberman. The dynamics of viral marketing. In *In EC 06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 228–237. ACM Press, 2005.

[174] Jianzhong Li, Yong Liu, and Hong Gao. Efficient algorithms for summarizing graph patterns. *IEEE Transactions On Knowledge And Data Engineering*, 23(9), 2011.

[175] Shao Li, Lijiang Wu, and Zhongqi Zhang. Constructing biological networks through combined literature mining and microarray analysis: a lmma approach. *Journal of Bioinformatics*, 2006.

[176] Yifan Li, Jiawei Han, and Jiong Yang. Clustering moving objects. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[177] Yong Liu, Jianzhong Li, and Hong Gao. Jpminer: Mining frequent jump patterns from graph databases. In *Proceedings of Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, 2009.

[178] Robert Lund and Jaxk Reeves. Detection of undocumented changepoints: a revision of the two-phase regression model. *J. Climate*, 2002.

[179] D. MacKay. Introduction to monte carlo methods. *Learning in graphical models*, 1998.

[180] Brandon Malone and Changhe Yuan. A parallel, anytime, bounded error algorithm for exact bayesian network structure learning. In *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM-12)*.

[181] D. Margaritis. Distribution-free learning of bayesian network structure in continuous domains. In *Proceedings of 2005 AAAI*, 2005.

[182] D. Margaritis and F. Bromberg. Efficient markov network discovery using particle filter. *Comp. Intel.*, 25(4):367C394, 2009.

[183] Paloma Mas. Circadian clock function in arabidopsis thaliana: time beyond transcription. *Trends Cell Biology*, 18:273–181, 2008.

[184] Claire Mathieu, Ocan Sankur, and Warren Schudy. Online correlation clustering. *arXiv:1001.0920v2*, 2010.

[185] Harley H. McAdams and Adam Arkin. Stochastic mechanisms in gene expression. *Proceeding of the National Academy of Science*, 94(3):814–819, 1997.

[186] A. McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2003.

[187] M. Meila and J. Shi. A random walks view of spectral segmentation. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, 2001.

[188] T. Minka. Algorithms for maximum-likelihood logistic regression. *Technical report*, 2001.

[189] T. Minka. Power ep. *Technical Report MSR-TR-2004-149, Microsoft Research*, 2004.

[190] Takeshi Mizuno and Norihito Nakamichi. Pseudo-response regulators (prrs) or true oscillator components (tocs). *Plant Cell Physiol.*, 46(5):677–685, 2005.

[191] S. Monti and G. F. Cooper. Learning bayesian belief networks with neural network estimators. In *Advances in Neural Information Processing Systems 9 (NIPS'96)*, 1996.

[192] V. Moskvina and A. Zhigljavsky. An algorithm based on singular spectrum analysis for changepoint detection. *Communications in Statistics Part B: Simulation and Computation*, 2003.

[193] Kevin Murphy and Saira Mian. Modeling gene expression data using dynamic bayesian networks. *Technical Report*, 1999.

[194] Mohamadreza Najiminaini, Laxmi Subedi, and Ljiljana Trajkovic. Analysis of internet topologies: A historical view. In *Proceedings of 2009 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1697 – 1700, 2009.

[195] N. Nariai, Sun Yong Kim, Seiya Imoto, and Satoru Miyano. Using protein-protein interactions for refining gene networks estimated from microarray data by bayesian networks. *Pacific Symposium on Biocomputing*, 9:336–347, 2004.

[196] Maria C.V. Nascimento and Andre C.P.L.F. de Carvalho. Spectral methods for graph clustering - a survey. *European Journal of Operational Research*, 211(2):221–231, 2011.

[197] Radford M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

[198] M.E.J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review*, 74:036C104, 2006.

[199] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, 69, 2004.

[200] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14:849C856, 2002.

[201] S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the 2004 SIGKDD Conference*, page 647C652, 2004.

[202] S. Nijssen and J.N. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 647–652, 2004.

[203] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas Huang. Incremental spectral clustering with application to monitoring of evolving blog communities. In *SIAM International Conference on Data Mining*, 2007.

[204] Agostino Nobile and Alastair T. Fearnside. Bayesian finite mixtures with an unknown number of components: The allocation sampler. *Statistics and Computing*, 17:147–162, 2007.

[205] C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. CATH - a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

[206] Larry Page. Pagerank: Bringing order to the web. Technical report, 1997.

[207] P. Pakzad and V. Anantharam. Belief propagation and statistical physics. In *Proceedings of the conference on Information Science and Systems*, 2002.

[208] Alessia Para, Eva M. Farre, Takato Imaizumi, Jose L. Pruneda-Paz, Franklin G. Harmon, and Steve A. Kay. Prr3 is a vascular regulator of toc1 stability in the arabidopsis circadian clock. *Plant Cell*, 19(11):3462–73, 2007.

[209] J. Pearl. A constraint-propagation approach to probabilistic reasoning. *Uncertainty in Artificial Intelligence*, pages 3718–382, 1986.

[210] J. Pearl. Fusion, propagation and structuring in belief networks. *UCLA Computer Science Department Technical Report*, 29(3):241–288, 1986.

[211] J. Pearl. Efficient algorithms for summarizing graph patterns. *Morgan Kaufmann*, 1988.

[212] J. Pearl. Probabilistic reasoning in intelligent systems. 1988.

[213] Judea Pearl. Heuristics: Intelligent search strategies for computer problem solving. *Addison-Wesley*, 1983.

[214] T. Penzl. Numerical solution of generalized lyapunov equations. *Advances in Comp. Math.*, 8:33–48, 1998.

[215] A. Pievatolo and Peter J. Green. Bounary detection through dynamic polygons. *Journal of the Royal Statistical Society*, Series B 60:609–626, 1998.

[216] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[217] L. De Raedt and S. Kramer. The levelwise version space algorithm and its application to molecular fragment finding. *In IJCAI'01: Seventeenth International Joint Conference on Artificial Intelligence*, 2:853–859, 2001.

[218] Pradeep Rai and Shubha Singh. A survey of clustering techniques. *International Journal of Computer Applications*, 7(12), 2010.

[219] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association (American Statistical Association)*, 66(336):846–850, 1971.

[220] A. Ranganathan. The dirichlet process mixture (dpm) model. *Technical report, Georgia Institute of Technology*, 2006.

[221] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, , and Christos Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *Proceedings of the 13th Intl. 4 Symposium on High-Performance Computer Architecture (HPCA).*

[222] Arvind Rao, Alfred O. Hero III, David J. States, and James Douglas Engel. Inferring time-varying network topologies from gene expression data. *EURASIP Journal on Bioinformatics and Systems Biology*, 2007.

[223] P. Ravikumar, M. J.Wainwright, and J. D. Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *Annals of Statistics*, 38:1287C1319, 2010.

[224] Sobia Raza, Kevin A Robertson, Paul A Lacaze, David Page, Anton J Enright, Peter Ghazal, and Tom C Freeman. A logic-based diagram of signalling pathways central to macrophage activation. *BMC System Biology*, 2:Article 36, 2008.

[225] Paul Resnick and Hal R. Varian. Recommender systems. In *Communications of the ACM*, volume 40(3), 1997.

[226] J. Rissanen. Stochastic complexity in statistical inquiry. *World Scientific Publishing Company*, 1989.

[227] Joshua W Robinson and Alexander J Hartemink. Non-stationary dynamic bayesian networks. *Procedding of Advances in Neural Information Processing Systems Conference*, 2008.

[228] Patrice A. Salome and C. Robertson McClung. The arabidopsis thaliana clock. *Journal of Biological Rhythms*, 19(5):425–435, 2004.

[229] Thomas Sandmann, Lars J. Jensen, Janus S. Jakobsen, Michal M. Karzynski, Michael P. Eichenlaub, Peer Bork, and Eileen E.M. Furlong. Da temporal map of transcription factor activity: mef2 directly regulates target genes at all stages of muscle development. *Dev Cell*, 10(6):797–807, 2006 Jun.

[230] E. J. Santos and S. E. Shimony. Deterministic approximation of marginal probabilities

in bayes nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(4):377–393, 1998.

[231] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[232] Federico Schluter. A survey on independence-based markov networks learning. *arXiv:1108.2283*, 2011.

[233] R. Shachter. Intelligent probabilistic inference. *Uncertainty in Artificial Intelligence*, pages 371–382, 1986.

[234] R. Shachter. Evidence absorption and propagation through evidence reversals. *Uncertainty in Artificial Intelligence*, pages 173–190, 1990.

[235] Yi Shan, Bo Wang, Jing Yan, Yu Wang, Ningyi Xu, and Huazhong Yang. Fpmr: Mapreduce framework on fpga a case study of rankboost acceleration. In *Proceedings of the FPGA'10*.

[236] Han Lin Shang. A survey of functional principal component analysis. Technical report, 2010.

[237] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[238] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on PAMI*, 22(8):888, 2000.

[239] Sandip Sinharay. Model diagnostics for bayesian networks. *Journal of Educational and Behavioral Statistics*, 31:1C33, 2006.

[240] Le Song, Mladen Kolar, and Eric Xing. Time-varying dynamic bayesian networks. *NIPS*, 2009.

[241] D.A. Spielman and S.H. Teng. Nearlylinear time algorithms for graph partitioning, graph sparsification, and solving linear systems. Proceedings of the Thirtysixth Annual Symposium on Theory of Computing, 2004.

[242] P. Spirtes, C. Glymour, and R. Scheines. Causation, prediction, and search. *Adaptive Computation and Machine Learning Series*, 2000.

[243] E. Steele, A. Tucker, P.A.C. 't Hoen, and M.J. Schuemie. Literature-based priors for gene regulatory networks. *Journal of Bioinformatics*, 2009.

[244] G.W. Stewart and J.G. Sun. Matrix pertubation theory. *Academic Press*, 1990.

[245] Laxmi Subedi and Ljiljana Trajkovic. Spectral analysis of internet topology graphs. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1803 – 1806, 2010.

[246] Yoshinori Tamada, Seiya Imoto, and Satoru Miyano. Parallel algorithm for learening optimal bayesian network structure. *Journal of Machine Learning Research*, 12:2437C2459, 2011.

[247] Lei Tang, Huan Liu, Jianping Zhang, and Zohreh Nazeri. Community evolution in dynamic multi-mode networks. In *Proceedings of the 2008 annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[248] Christoffer Valgren, Tom duckett, and Achim Lilienthal. Incremental spectral clustering and its application to topological mapping. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2007.

[249] S.M. van Dongen. Graph clustering by flow simulation. *Ph.D. Thesis*, 2000.

[250] R. A. van Engelen. Approximating bayesian belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):916– 920, 1997.

[251] S. V. N. Vishwanathan, N. N. Schraudolph, M.W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd international conference on Machine learning*, page 969C976, 2006.

[252] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, 2009.

[253] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[254] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In *Proceedings of 2003 AISTATS*, 2003.

[255] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Machine Learning*, 1(1-2):1–305, 2008.

[256] G. Wang, R. L. Dunbrack, and Jr. PISCES. A protein sequence culling server. *Bioinformatics*, 19:1589–1591, 2003.

[257] H. Wang, K. Yu, and H. Yao. Learning dynamic bayesian networks using evolutionary mcmc. In *Proceedings of the International Conference on Computational Intelligence and Security*, pages 45–50, 2006.

[258] J. Wang and J. Han. Bide: Efficient mining of frequent closed sequences. In *Proceeding of the 2004 international conference on data engineering (ICDE04)*, page 79C90, 2004.

[259] Y. Weiss and W. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. In *Proceedings of NIPS'99*, 1999.

[260] Cheng-Hsiung Weng and Yen-Liang Chen. Mining fuzzy association rules from uncertain data. *Knowledge and Information Systems*, 23(2):129–152, 2010.

[261] Adriano V Werhli, , and Dirk Husmeier. Gene regulatory network reconstruction by bayesian integration of prior knowledge and/or dirent experimental conditions. *Journal of Bioinformatics and Computational Biology*, 6(3):543–572, 2008.

[262] Adriano V. Werhli, Marco Grzegorczyk, and Dirk Husmeier. Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and bayesian networks. *Bioinformatics*, 20(22):2523–2531, 2006.

[263] Scott White and Padhraic Smyth. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 2005 SIAM Data Mining Conference*, 2005.

[264] J. H. Wilkinson. The algebraic eigenvalue problem. *Clarenon Press*, 1965.

[265] J. Winn and C. M. Bishop. Variational message passing. *J. Mach. Learn. Res.*, 6:661C694, 2005.

[266] Kein S. Xu, Mark Kliger, and Alfred O. Hero III. Adaptive evolutionary clustering. *arXiv:1104.1990v1*, 2011.

[267] Kevin S. Xu, Mark Kliger, and Algred O. Hero III. Evolutionary spectral clustering with adaptive forgetting factor. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010.

[268] Kuai Xu, Feng Wang, and Lin Gu. Network-aware behavior clustering of internet end hosts. In *Proceedings of IEEE INFOCOM 2011*, pages 2078 – 2086, 2011.

[269] Katsutoshi Yada, Hiroshi Motoda, Takashi Washio, and Asuka Miyawaki. Consumer behavior analysis by graph mining technique. *Lecture Notes in Computer Science*, pages 800–806, 2004.

[270] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Procceeding of International Conference on Data Mining (ICDM'02)*, pages 721–724, 2002.

[271] X. Yan and J. Han. Closegraph: Mining closed frequentgraph patterns. In *Proc. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 03)*, pages 286–295, 2003.

[272] Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large datasets. In *Proceedings of 2003 SDM Conference*, pages 166–177, 2003.

[273] Xifeng Yan, Feida Zhu, Philip S. Yu, and Jiawei Han. Feature-based substructure similarity search. *ACM Transactions on Database Systems*, 31(4):1418–1453, 2006.

[274] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7), 2005.

[275] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Proceedings of Thirteenth NIPS*, 2001.

[276] Richard M. Yoo, Anthony Romano, and Christos Kozyrakis. Phoenix rebirth: Scalable mapreduce on a large-scale shared-memory system. In *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*.

[277] Haiyuan Yu, Nicholas M. Luscombe, Jiang Qian, and Mark Gerstein. Genomic analysis of gene expression relationships in transcriptional regulatory networks. *Trends Genet*, 19:422–7, 2003.

[278] Jing Yu, V. Anne Smith, Paul P. Wang, Alexander J. Hartemink, and Erich D. Jarvis. Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20:3594–3603, 2004.

[279] Changhe Yuan and Brandon Malone. An improved admissible heuristic for learning optimal bayesian networks. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI-12)*, Catalina Island, CA, 2012.

[280] N. L. Zhang and D. Poole. A simple approach to bayesian network computations. In *Proc. of the Tenth Canadian Conference on Artificial Intelligence*, pages 171–178, 1994.

[281] Shijie Zhang and Jiong Yang. Ram: Randomized approximate graph mining export. *Scientific and Statistical Database Management*, 2008.

[282] Shijie Zhang, Jiong Yang, and V. Cheedella. Monkey: Approximate graph mining based on spanning trees. In *Proceeding of IEEE 23rd International ConferenceData Engineering (ICDE'07)*, pages 1247–1249, 2007.

[283] Wentao Zhao, Erchin Serpedin, and Edward R. Dougherty. Inferring gene regulatory networks from time series data using the minimum description length principle. *Bioinformatics*, 22(17):2129–2135, 2006.

[284] Wentao Zhao, Erchin serpedin, and Edward R. Dougherty. Inferring gene regulatory networks from time series data using the minimum description length principle. *Bioinformatics*, 22(17):2129–2135, 2006.

[285] V. Ziegler. Approximation algorithms for restricted bayesian network structures. *Information Processing Letters*, 108(2):60C63, 2008.

[286] Min Zou and Suzanne D. Conzen. A new dynamic bayesian network (dbn) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21:71–79, 2004.

[287] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Frequent subgraph pattern

mining on uncertain graph data. In *Proceedings of the 2009 Conference on Information and Knowledge Management (CIKM'09)*, pages 583–592, 2009.