Feedback Message Passing for Inference in Gaussian Graphical Models

by

Ying Liu

B.E. Electrical Engineering
Tsinghua University, Beijing, 2008

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2010

Signature of Author .....
Department of Electrical Engineering and Computer Science
May 7, 2010

Certified by .....
Alan S. Willsky
Edwin Sibley Webster Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Feedback Message Passing for Inference in Gaussian Graphical Models

by

Ying Liu

Submitted to the Department of Electrical Engineering and Computer Science
on May 7, 2010, in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

For Gaussian graphical models with cycles, loopy belief propagation often performs reasonably well, but its convergence is not guaranteed and the computation of variances is generally incorrect. In this paper, we identify a set of special vertices called a feedback vertex set whose removal results in a cycle-free graph. We propose a *feedback message passing* algorithm in which non-feedback nodes send out one set of messages while the feedback nodes use a different message update scheme. Exact inference results can be obtained in $\mathcal{O}(k^2n)$, where k is the number of feedback nodes and n is the total number of nodes. For graphs with large feedback vertex sets, we describe a tractable approximate feedback message passing algorithm. Experimental results show that this procedure converges more often, faster, and provides better results than loopy belief propagation.

Thesis Supervisor: Alan S. Willsky
Title: Edwin Sibley Webster Professor of Electrical Engineering

Acknowledgments

I am extremely fortunate to work under the supervision of Professor Alan Willsky. His incredible deep and extensive knowledge has been a great inspiration to me. He has the remarkable ability to explain even the most difficult concepts in a few words any time. Alan is very supportive to all my research ideas. During our discussions, he proposed many interesting and challenging questions, which have been amazing inputs to my research. Alan always quickly gives me feedback of our papers and this thesis after careful review. Thanks, Alan, for many things.

It is really my pleasure to work in the stochastic systems group with so many wonderful people. Venkat Chandrasekaran taught me walk-sum analysis and patiently answered many of my questions. He is always ready to talk with me about any research ideas, many of which became the basis of this thesis. I have also enjoyed collaborating with Anima Anandkumar, who introduced many helpful resources to me and spent time brainstorming with me. Justin Dauwels has been very helpful to me. He carefully reviewed my thesis chapters and gave me many useful comments. Justin has provided many great insights for my research. Discussing research ideas with him has been a very pleasant time. I spent wonderful time with Matt Johnson, who came to the group the same time with me, taking classes, attending talks and working out. Also I thank Jason Chang, Myung Jin Choi, Emily Fox, Dahua Lin, James Saunderson, Vincent Tan and Kush Varshney, for making the group as great as it is.

I thank many professors in MIT, particularly my academic advisor Alan Oppenheim, who put great efforts in preparing for the lectures and spent time talking with me.

I am also grateful to a lot of friends in and outside MIT, especially Ramesh Sridharan and Roger Grosse, for attending many interesting events with me during the years. Thanks to all my friends who chatted, had dinners, played games, traveled with me.

Above all, I thank my parents Xifan Liu and Xiurun Yu for their love. I have been studying far away from them since my first year in high school. As their only child, I have not been able to visit them as often as I should. Thanks, for everything. You being happy and healthy is the biggest support to me.

Contents

List of Figures

Chapter 1

Introduction

In graphical models each node represents a random variable and the edge structure specifies the conditional independence or Markov properties of the underlying distribution [24]. In Gaussian graphical models, the distributions are Gaussian distributions, which have desirable analytical convenience. Such models are widely used in many fields such as computer vision [30,36], gene regulatory networks [3,43,44], medical diagnostics [5,17], oceanography [13,45], and communication systems [11,21].

Although inference in Gaussian graphical models can be solved by direct matrix inversion for problems of moderate sizes, it is intractable for very large problems involving millions of random variables [15,45,46]. Efficient algorithms for solving such inference problems are of great importance.

1.1 Motivations for the Feedback Message Passing Algorithm

Belief propagation (BP) is an efficient message passing algorithm that gives exact inference results in linear time for tree-structured graphs. The Kalman filtering algorithm in linear estimation and forward-backward method in hidden Markov chain, can be viewed as special cases of belief propagation. However, tree-structured models possess limited modeling

capabilities, and many real world processes cannot be well-modeled using graphs without cycles.

For inference in loopy graphs, loopy belief propagation (LBP) can be used as a direct extension of BP by following the same local message passing rules. It turns out that LBP performs reasonably well for certain loopy graphs [8,32]. The famous Turbo coding method is also shown to be a successful example of LBP [31].

However, the convergence and correctness of LBP are not guaranteed in general, and many studies have been conducted on the performance of LBP [20,29,38,42]. For Gaussian graphical models, if LBP converges, the means converge to the correct values while the variances are generally incorrect [42]. In [29] a walk-sum analysis framework is proposed to analyze the performance of LBP in Gaussian graphical models. Based on the walk-sum analysis some algorithms are proposed to obtain better inference results [6,22,37].

A desirable property of LBP is that it is completely distributed. Any message update only involves local information and local messages so that each node can do the computations simultaneously. However, LBP has its limitations: global information of the cyclic structure of the graph is not captured and thus errors and convergence problems may occur. One can ask some natural questions: can we use some more memory to keep track of the messages or use some header information to denote the sources of the messages? Are there some nodes that are more important in terms of inference? Can we design an algorithm accordingly without losing too much decentralization?

We consider a particular set of "important" nodes called *feedback vertex set* (FVS). A feedback vertex set is a subset of vertices that breaks all the cycles in the graph. The high level idea of our algorithm is to give priority to the "important" nodes, which use a different message passing scheme than other nodes. Based on this viewpoint, we propose an algorithm for Gaussian graphical models. The algorithm includes several message passing steps. The whole procedure takes linear time to obtain the exact means and variances for all nodes if the number of feedback nodes is bounded by a constant. More generally, the complexity is $\mathcal{O}(k^2 n)$, where k is the number of the feedback nodes and n is the total number of nodes. For

graphs with large feedback vertex sets, the exact feedback message passing can be intractable. For such situations we develop an approximate feedback message passing algorithm to obtain approximate results, where we can trade off between efficiency and accuracy explicitly.

1.2 Thesis Overview

The remainder of the thesis is organized as follows.

Chapter 2. Background

We first introduce some basic concepts in graph theory, graphical models and Gaussian graphical models. Then we describe belief propagation and its immediate extension to loopy belief propagation. A walk-sum analysis interpretation of inference problems and algorithms is then discussed. Finally we introduce the key concept of feedback vertex set and some relevant results and algorithms.

Chapter 3. Exact Feedback Message Passing

We show that for a class of graphs with small feedback vertex set, inference problems can be solved efficiently by the feedback message passing algorithm. We start with the single feedback node case, and illustrate the algorithm by concrete examples. Then we describe the general algorithm with multiple feedback nodes, which is a direct extension of the single feedback node case. We also prove the convergence and correctness of the algorithm.

Chapter 4. Approximate Feedback Message Passing

Though the exact feedback message passing algorithm always gives correct inference results, it can be intractable when the size of the FVS is large. In this case, we can make some approximations to obtain tractable algorithms. One natural way to make an approximation is to use a set of feedback nodes of bounded size. Then we perform inference among the non-feedback nodes approximately (using loopy belief propagation) while solving the small

inference problem on the set of feedback nodes exactly. In this chapter we describe this approximate algorithm and give relevant theoretical results.

Chapter 5. Numerical Results

We give numerical results of the performance of our algorithms. The experiments are performed on grid graphs, in which inference problems are difficult to solve. We design a series of experiments to analyze the convergence and accuracy of the approximate feedback message passing algorithm. We also compare the performance of the algorithm with different choices of pseudo-FVS.

Chapter 6. Conclusions

We summarize our results and propose future research problems.

Chapter 2

Background

2.1 Graphical Models

A probability distribution of multiple random variables can be a complicated multivariate function in general. However, there is often structural information that can be exploited. As an extreme example, if the random variables are mutually independent, the density function can be factorized into a product of univariate functions. A rich class of distributions have conditional independence properties, which convey structural information implicitly. A graphical model is a convenient and intuitive representation of such structural information, where conditional independence properties can be read explicitly from the graph structure.

2.1.1 Basics in Graph Theory

Here we introduce some basic notions in graph theory that will be used throughout this thesis. A comprehensive introduction to graph theory can be found in [10].

A *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of *nodes* or *vertices* $\mathcal{V}$ and a set of *edges* $\mathcal{E}$. The edges are pairs of two distinct nodes in $\mathcal{V}$. These pairs can be ordered or unordered pairs. A graph is a *directed graph* if the edges are ordered pairs or an *undirected graph* if the edges are unordered pairs. In this thesis, we focus exclusively on undirected graphs. An edge is represented by (s, t), where s and t are distinct nodes in $\mathcal{V}$. We also say node s and node t

are *connected* by edge (s,t). A graph is called a *complete* or *fully connected graph* if any two nodes are connected.

The set of *neighbors* of node i is the set of nodes connected to node i, which is denoted by $\mathcal{N}(i)$. The *degree* of node i is the number of its neighbors, which is denoted by $\deg(i)$. A *walk* on a graph is a finite or infinite sequence of nodes in which the consecutive nodes are neighbors in the graph. A *path* is a finite walk with distinct nodes. A graph is called a *connected graph* if there exists a path between any pair of nodes. A *cycle* or *loop* is a walk that starts and ends at the same node but all other nodes are distinct. A *forest* is a graph without cycles. If a forest is a connected graph, it is called a *tree*. In this thesis, we use *tree-structured graph* to refer to forests as well.

A *subgraph* $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph satisfying $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$, where $\mathcal{E}'$ only involves nodes in $\mathcal{V}'$. A *clique* in a graph is a fully connected subgraph. A *maximal clique* is a clique that is not contained in any larger clique.

Let A, B and C be three sets of nodes in a graph. Set C *separates* set A and set B if any path from a node in A and a node in B contains at least one node in C.

2.1.2 Graph Representation of Probability Distributions

Graphical models defined on undirected graphs are called undirected graphical models, which are also known as *Markov graphical models* or *Markov random fields*.

Consider a probability distribution $p(x_1, x_2, \cdots, x_n)$ of n random variables. We use an undirected graph with n nodes to represent the conditional independence structure, where each node corresponds to a random variable. The edges of the graph are constructed such that the distribution is *Markov with respect to the graph*: if set A and set B are separated by set C, $\mathbf{x}_A$ and $\mathbf{x}_B$ are independent conditioned on $\mathbf{x}_C$, namely, $p(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C) = p(\mathbf{x}_A|\mathbf{x}_C)p(\mathbf{x}_A|\mathbf{x}_C)$. Based on this graph representation, many conditional independence properties can be read immediately from the graph. For example, in Figure 2-1 node e is independent of node b and c conditioned on a and d, since set $\{a, d\}$ separates node e from set $\{b, c\}$ in the graph. Roughly speaking, any missing edge from a complete graph can give

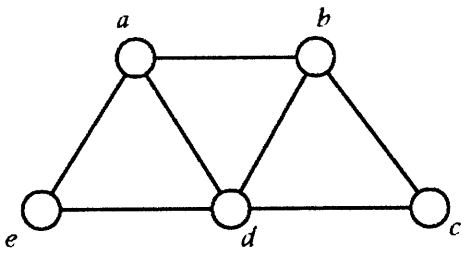


Figure 2-1: A simple Markov graphical model

us some structural information about the distribution.

The graph structure of a graphical model can lead to a factorization of the probability distribution function. By the Hammersley-Clifford Theorem: If a probability distribution $p(x_1, x_2, \cdots, x_n) > 0, \forall \mathbf{x} = (x_1, x_2, \cdots, x_n)$, then $p(\mathbf{x})$ can be factorized according to the underlying graph structure by

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C), \tag{2.1}$$

where $\mathcal{C}$ is the set of all maximal cliques in the graph, $\psi_C(\mathbf{x}_C)$ is a positive function defined on $\mathbf{x}_C$, and $Z = \sum_{\mathbf{x}} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C)$ is the normalization factor, which is often called *partition function* [24].

For a class of *pairwise Markov graphical models*, the functions defined on the cliques can be further factorized into products of bivariate functions, corresponding to nodes or edges on the graph. For such models, the distribution can be factorized as

$$p(x_1, x_2, \cdots, x_n) = \frac{1}{Z} \prod_{l \in \mathcal{V}} \phi(x_l) \prod_{(i,j) \in \mathcal{E}} \phi(x_i, x_j), \tag{2.2}$$

where Z is the normalization factor.

Inference problems are to extract useful information from a given graphical model. In this

thesis we are most interested in computing the marginal distribution of each node. Namely,

$$p_i(x_i) = \sum_{x_j \in \mathcal{V}, j \neq i} p(x_1, x_2, \cdots, x_n), \ \forall i \in \mathcal{V}. \tag{2.3}$$

See [24] for detailed introduction to graphical models.

2.2 Gaussian Graphical Models

Exponential families of probability distributions have been widely studied [16, 18, 40]. The distributions in this family are in the form

$$p(\mathbf{x}; \boldsymbol{\theta}) = \exp\{\sum_a \theta_a \phi_a(\mathbf{x}_a) - \Phi(\boldsymbol{\theta})\}, \tag{2.4}$$

where $\phi_a(x)$ are *sufficient statistics*, θ_a are *natural parameters*, and

$$\Phi(\boldsymbol{\theta}) = \log \int \exp(\sum_a \theta_a \phi_a(\mathbf{x}_a)) \mathrm{d}\mathbf{x} \tag{2.5}$$

is the *log partition function.*

The domain of the natural parameters is the set of parameters that yield a *normalizable distribution* or *valid distribution*, namely $\Theta = \{\boldsymbol{\theta} | \Phi(\boldsymbol{\theta}) < \infty\}$.

Under the graphical model framework, $\phi_a(\mathbf{x}_a)$ can be viewed as a function of a clique. Therefore, we can construct the graph structure of a graphical model according to the given sufficient statistics.

It is not hard to show that if we take the derivatives of the log partition function $\Phi(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, we can obtain the moments of $\phi_a(\mathbf{x}_a)$:

$$\frac{\partial \Phi}{\partial \theta_a}(\boldsymbol{\theta}) = \mathbb{E}\{\phi_a(\mathbf{x}_a)\} \tag{2.6}$$

$$\frac{\partial^2 \Phi}{\partial \theta_a \partial \theta_b}(\theta) = \mathbb{E}\{(\phi_a(\mathbf{x}_a) - \mathbb{E}\{\phi_a(\mathbf{x}_a)\})(\phi_b(\mathbf{x}_b) - \mathbb{E}\{\phi_b(\mathbf{x}_b)\})\}, \tag{2.7}$$

where the expectation is taken with respect to $p(\mathbf{x};\boldsymbol{\theta})$. For a valid distribution, the covariance matrix is positive semidefinite, and thus the log partition function is a convex function of θ within the domain.

An important exponential family of distributions is the family of Gaussian distributions. In Gaussian graphical models, the probability distributions are given by $p(\mathbf{x}) \propto \exp\{-\frac{1}{2}\mathbf{x}^T J\mathbf{x} + \mathbf{h}^T\mathbf{x}\}$, where J is called the *information, concentration or precision matrix* and $\mathbf{h}$ is called the *potential vector.*

We can define an underlying graph for the model, which is sparse with respect to J: $\forall(i,j) \notin \mathcal{E}$, $J_{ij} = 0$. For a valid Gaussian distribution, J is symmetric and positive definite. Compared to (2.4) we can relate the parameters of Gaussian distributions to the parameters of the exponential family by:

$$\begin{aligned}\{\theta_a\} &= \{h_i\} \cup \{-\frac{1}{2}J_{ii}\} \cup \{-J_{ij}\},\ \forall i \in \mathcal{V},\ \forall(i,j) \in \mathcal{E},\\ \{\phi_a(\mathbf{x}_a)\} &= \{x_i\} \cup \{x_i^2\} \cup \{x_i x_j\},\ \forall i \in \mathcal{V},\ \forall(i,j) \in \mathcal{E}.\end{aligned}$$

Conditional independence in Gaussian distributions can be read immediately from the underlying graph structure. We can also directly read the conditional independence from the sparsity pattern of the information matrix. For instance, if $J_{ij} = 0, i \neq j$, then x_i and x_j are independent conditioned on all other variables. In general if $J = \begin{bmatrix} J_{AA} & J_{AB} \\ J_{BA} & J_{BB} \end{bmatrix}$ and $\mathbf{h} = \begin{bmatrix} \mathbf{h}_A \\ \mathbf{h}_B \end{bmatrix}$, where A and B are sets of indices, the conditional distribution of $\mathbf{x}_A$ given $\mathbf{x}_B = \tilde{\mathbf{x}}_B$ has information matrix $J_{A|B} = J_{AA}$ and potential vector $\mathbf{h}_{A|B} = \mathbf{h}_A - J_{AB}\tilde{\mathbf{x}}_B$. Moreover, the conditional variance of x_i given its neighbors $\mathbf{x}_{\mathcal{N}(i)}$ is $\mathrm{Var}\{x_i|\mathbf{x}_{\mathcal{N}(i)}\} = (J_{ii})^{-1}$. The marginal distribution of $\mathbf{x}_A$ has the information matrix $\hat{J}_{AA}$ given by $\hat{J}_{AA} = J_A - J_{AB}J_{BB}^{-1}J_{BA}$ and potential vector $\hat{\mathbf{h}}_A = \mathbf{h}_A - J_{AB}J_{BB}^{-1}\mathbf{h}_B$.

The relationship with the mean $\boldsymbol{\mu} = \mathbb{E}\{\mathbf{x}\}$ and the covariance matrix $P = \mathbb{E}\{(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\}$ is $\boldsymbol{\mu} = J^{-1}\mathbf{h}$ and $P = J^{-1}$.

2.3 Belief Propagation

Belief propagation (BP) is a distributed message passing algorithm for solving inference problems in graphical models, where each message is updated locally [33]. There are several versions of BP such as sum-product and max-product for different inference problems. In this thesis we use the sum-product version for obtaining marginal distributions. BP is well defined on both tree-structured graphs and loopy graphs. The latter is often referred as loopy belief propagation (LBP). The convergence and correctness of BP are guaranteed for tree-structured graph. However, they are not guaranteed for LBP in general.

2.3.1 Belief Propagation for Tree-Structured Graphs

BP on a tree-structured graph gives the exact marginal distributions in linear time. In the asynchronous version, message $m_{i\to j}$ is sent from node i to its neighbor j after node i receives messages from all other neighbors, where a suitable message scheduling scheme is needed[1]. Alternatively, we can use the synchronous version of BP, where messages are updated in an iterative way. In each iteration all messages are updated according to previous messages. After convergence, all the messages reach the fixed points. Then we can use the fixed point messages to compute the marginals and pairwise marginals for all nodes and edges.

Here is a summary of the synchronous version of belief propagation on tree-structured graphical models. Consider a graphical model

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \psi_i(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j). \tag{2.8}$$

1. **Message Passing**

   We initialize all the messages as $m_{i\to j}^{(0)} = 1$, $\forall (i,j) \in \mathcal{E}$.

[1]Start from the leaf nodes moving toward common "root" and then return to the leaves.

At each iteration, the messages are updated based on previous messages:

$$m_{i\to j}^{(t)}(x_j) = \int \psi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{k\in\mathcal{N}(i)\setminus j} m_{k\to i}^{(t-1)}(x_i)\mathrm{d}x_i, \ \ \forall (i,j)\in\mathcal{E}. \tag{2.9}$$

After convergence, all the messages satisfy the fixed point equations:

$$m_{i\to j}(x_j) = \int \psi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{k\in\mathcal{N}(i)\setminus j} m_{k\to i}(x_i)\mathrm{d}x_i, \ \ \forall (i,j)\in\mathcal{E}. \tag{2.10}$$

For tree-structured models, the fixed point messages are obtained in linear time for any initial values.

2. **Marginals Computation**

The marginals and pairwise marginals are computed based on the fixed point messages.

$$p_i(x_i) = \frac{1}{Z_i}\psi_i(x_i) \prod_{k\in\mathcal{N}(i)} m_{k\to i}(x_i), \tag{2.11}$$

where Z_i is the normalization factor.

$$p_{ij}(x_i,x_j) = \frac{1}{Z_{ij}}\psi_i(x_i)\psi_j(x_j)\psi_{ij}(x_i,x_j) \prod_{k\in\mathcal{N}(i)\setminus j} m_{k\to i}(x_i) \prod_{l\in\mathcal{N}(j)\setminus i} m_{l\to j}(x_j), \tag{2.12}$$

where Z_{ij} is the normalization factor.

For tree-structured graph, the marginals and pairwise marginals computed above are exact.

In practice, each message can be represented by a vector for discrete models. For continuous models, each BP message is a real function which is difficult to store and transfer precisely. However, in Gaussian graphical models, the messages are Gaussian functions, which can be easily represented by a few parameters. Here we also give the Gaussian BP as

a special case of BP. Consider a Gaussian graphical model

$$p(\mathbf{x}) \propto e^{-\frac{1}{2}\mathbf{x}^T J\mathbf{x}+\mathbf{h}^T\mathbf{x}}. \tag{2.13}$$

1. **Message Passing**

   We initialize all the messages as $\Delta J_{i\to j}^{(0)}$ and $\Delta h_{i\to j}^{(0)}$, $\forall (i,j) \in \mathcal{E}$.

   At each iteration, the messages are updated based on previous messages:

$$\Delta J_{i\to j}^{(t)} = -J_{ji}(\hat{J}_{i\backslash j}^{(t-1)})^{-1}J_{ij} \text{ and } \Delta h_{i\to j}^{(t)} = -J_{ji}(\hat{J}_{i\backslash j}^{(t-1)})^{-1}\hat{h}_{i\backslash j}^{(t-1)}, \tag{2.14}$$

   where

$$\hat{J}_{i\backslash j}^{(t-1)} = J_{ii} + \sum_{k\in\mathcal{N}(i)\backslash j} \Delta J_{k\to i}^{(t-1)} \text{ and } \hat{h}_{i\backslash j}^{(t-1)} = h_i + \sum_{k\in\mathcal{N}(i)\backslash j} \Delta h_{k\to i}^{(t-1)}. \tag{2.15}$$

   After convergence, all the messages reach the fixed points: $\Delta J_{i\to j}$ and $\Delta h_{i\to j}$, $\forall (i,j) \in \mathcal{E}$.

   For tree-structured models, the fixed point messages are obtained in linear time for any initial values.

2. **Marginals Computation**

   The variances, means and covariances for all nodes or edges are computed based on the fixed point messages:

$$\hat{J}_i = J_{ii} + \sum_{k\in\mathcal{N}(i)} \Delta J_{k\to i} \text{ and } \hat{h}_i = h_i + \sum_{k\in\mathcal{N}(i)} \Delta h_{k\to i}, \tag{2.16}$$

   which can be converted to the variance and mean by $P_{ii} = \hat{J}_i^{-1}$ and $\mu_i = \hat{J}_i^{-1}\hat{h}_i$.

   Every edge $(i,j) \in \mathcal{E}$ computes the covariance between x_i and x_j by

$$P_{ij} = J_{ij}/[J_{ij}^2 - (J_{ii} + \sum_{k\in\mathcal{N}(i)\backslash j} \Delta J_{k\to i})(J_{jj} + \sum_{l\in\mathcal{N}(j)\backslash i} \Delta J_{l\to j})]. \tag{2.17}$$

For tree-structured graph, the variances, means and covariances computed above are exact.

The convergence and correctness of BP on trees are given by the following proposition.

Proposition 2.1. *Belief propagation for tree-structured Gaussian graphical models is guaranteed to converge in linear time and gives exact variances and means for all nodes.*

Proof. See [24]. □

2.3.2 Belief Propagation for Loopy Graphs

LBP has the same message passing and marginals computation steps. In the message passing step, the messages are updated by the same equation as in (2.9). However, for general graphs, the messages are not guaranteed to reach fixed points. If the messages do reach fixed points, we proceed to compute the "marginals" and "pairwise marginals" as in (2.11) and (2.12). Note that even if the messages reach fixed points, the marginals and pairwise marginals we obtained are inaccurate in general.

For Gaussian graphical models, the messages are updated as in (2.14). When the messages reach fixed points, we compute the LBP approximations of the variances, means and covariances by (2.16) and (2.17). Again, the messages are not guaranteed to converge. Even if they converge, the estimated variances are not accurate in general.

An important property of LBP in Gaussian graphical models is as follows.

Proposition 2.2. *When LBP converges in Gaussian graphical models, i.e. all the messages reach fixed points, the means given by LBP are exact.*

Proof. Refer to Claim 5 in [42] for a proof. In [28] a walk-sum interpretation is provided in Proposition 3.2.4. □

2.4 Walk-sum Analysis

Inference in Gaussian graphical models can be interpreted from a walk-sum point of view. The variances and means can be viewed as the sums of walks defined on the underlying graph. Thus the algorithms for solving inference problems can be considered as capturing walks. Walk-sum analysis is useful because we can use this interpretation to verify and develop inference algorithms. If we can show that the walks captured by a certain algorithm are exactly all necessary walks, the correctness of the algorithm is then verified. On the other hand, we can also analyze why some algorithms do not give the accurate inference results by showing the missing walks.

For a Gaussian graphical model, we can normalize the information matrix J such that all the diagonal entries equal 1. Let $R = I - J$, then R has zero diagonal. Every non-zero off-diagonal entry corresponds to an edge in the underlying graph. We set the weight of the edge between node i and j as R_{ij}, the i, j^{th} entry of R. The matrix R is called the *edge weight matrix.*

A walk of length $l \geq 0$ is defined as a sequence $w = (w_0, w_1, w_2, ..., w_l)$, where $w_i \in \mathcal{V}$, $i \in 0, 1, ..., l$ and each step (w_i, w_{i+1}) corresponds to an edge in the graph. The *weight of a walk* is defined as the product of the edge weights,

$$\phi(w) = \prod_{k=1}^{l(w)} R_{w_{k-1}, w_k}, \tag{2.18}$$

where $l(w)$ is the length of walk w. Also, we define the weight of a zero length walk, i.e., a single node as 1.

By the Neumann power series for matrix inversion,

$$P \triangleq J^{-1} = (I - R)^{-1} = \sum_{k=0}^{\infty} R^k, \text{ for } \rho(R) < 1, \tag{2.19}$$

where $\rho(R)$ is the spectral radius of R, defined as the largest absolute eigenvalue for a symmetric matrix R.

We define the *walk-sum* or *sum of walks* as the sum of weights of walks in a set:

$$\phi(\mathcal{W}) \triangleq \sum_{w \in \mathcal{W}} \phi(w), \tag{2.20}$$

where $\mathcal{W}$ is a set of walks. We denote $\phi(i \to j)$ as the sum of all walks from node i to node j. Also, we call $\phi(i \to i)$ the *self-return walk-sum* of node i.

We can show that the i, j^{th} entry of R^l is the sum of all walks of length l from node i to node j, denoted by $\phi^l(i \to j)$. Then

$$P_{ij} = \sum_{l=0}^{\infty} \phi^l(i \to j). \tag{2.21}$$

Therefore, $P_{ij} = \phi(i \to j) = \sum_{l=0}^{\infty} \phi^l(i \to j)$.

A Gaussian graphical model is *walk-summable* if for all $i, j \in \mathcal{V}$, the walk-sum $\phi(i \to j)$ converges for any order of the summands. Several equivalent conditions for walk-summability are given as follows.

Proposition 2.3. *The following conditions are equivalent to walk-summability:*

(i) $\sum_{\{w: i \to j\}} |\phi(w)|$ *converges for all* $i, j \in \mathcal{V}$, *where* $\{w : i \to j\}$ *is the set of walks from* i *to* j.

(ii) $\rho(\bar{R}) < 1$, *where* $\bar{R}$ *takes the entrywise absolute value of* R.

Proof. See Proposition 3.1.1 in [28]. □

In walk-summable models, $\phi(i \to j)$ is well-defined for all $i, j \in \mathcal{V}$. The covariances and means can be expressed as

$$P_{ij} = \phi(i \to j) \quad \text{and} \quad \mu_i = \sum_{j \in \mathcal{V}} h_j P_{ij} = \sum_{j \in \mathcal{V}} h_j \phi(i \to j). \tag{2.22}$$

For walk-summable models, there are some important properties:

1. If $\{\mathcal{U}_n\}_{n=1}^{\infty}$ is a countable collection of mutually disjoint walks, we have $\phi(\bigcup_{n=1}^{\infty} \mathcal{U}_n) = \sum_{n=1}^{\infty} \phi(\mathcal{U}_n)$.

2. If $u = (u_0, ..., u_n)$, $v = (v_0, ...v_m)$ and $u_n = v_0$, we define the product of walks u and v as the concatenation of u and v, namely, $uv = (u_0, ..., u_n, v_1, ..., v_m)$. Consider a set $\mathcal{U}$ with all walks ending at node u_{end} and a set $\mathcal{V}$ with all walks starting at node $v_{start} = u_{end}$. We define $\mathcal{UV} = \{uv | u \in \mathcal{U}, v \in \mathcal{V}\}$. If $\forall w \in \mathcal{UV}$ there is a unique pair $(u, v) \in \mathcal{U} \times \mathcal{V}$ such that $uv = w$, we call $(\mathcal{U}, \mathcal{V})$ a valid decomposition. For any valid decomposition $(\mathcal{U}, \mathcal{V})$, $\phi(\mathcal{UV}) = \phi(\mathcal{U})\phi(\mathcal{V})$

By the above properties, we have two formulas as follows.

1. The *single-visit self-return walk-sum* $\phi(i \xrightarrow{\setminus i} i)$ denotes the walk-sum of all non-zero length walks that start at node i and end at node i but do not visit node i in between. We have

$$\phi(i \to i) = \frac{1}{1 - \phi(i \xrightarrow{\setminus i} i)}. \tag{2.23}$$

2. For $i \neq j$, the *single-visit walk-sum* $\phi(i \xrightarrow{\setminus j} j)$ denotes the walk-sum of all the walks that start at node i and end at node j but do not visit node j in between. We have

$$\phi(i \to j) = \phi(i \xrightarrow{\setminus j} j)\phi(j \to j). \tag{2.24}$$

A comprehensive walk-sum analysis of belief propagation in Gaussian graphical models is given in [28]. Here we list some propositions that we will use later in this thesis.

Proposition 2.4. *A Gaussian graphical model is walk-summable if it is attractive, i.e. any edge weight R_{ij}, is nonnegative.*

Proposition 2.5. *LBP converges for a walk-summable Gaussian graphical model.*

Proposition 2.6. *In walk-summable models, the LBP variance estimation for a node is a sum over all backtracking walks of the node, which is a subset of all self-return walks needed to calculate the correct variance.*

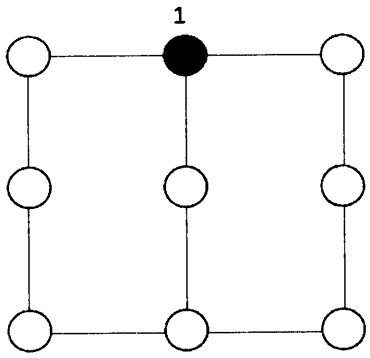


Figure 2-2: An example of the FVS

Here a backtracking walk is a self-return walk that can be reduced consecutively to a single node. Each reduction is to replace a subwalk of the form $\{i, j, i\}$ by the single node $\{i\}$.

2.5 Feedback Vertex Set

A *feedback vertex set* (FVS), sometimes also called a loop cutset or cutset, is defined as a set of vertices whose removal results in an acyclic graph [39]. For instance, node 1 in Fig 2-2 forms an FVS by itself. The problem of finding the FVS of the minimum size is called the *minimum feedback vertex set* problem, which has been widely studied in graph theory and computer science community. It has many applications such as genome assembly and VLSI chip design [4,34]. In combinatorial circuits, some circuit components may receive new input signals before they stabilize if there are cycles in the circuits. We can solve the problem by placing some clocked registers at the FVS so that the cyclic effect can be eliminated. Finding a minimum FVS is then a natural problem for minimizing the cost of registers.

For a general graph, the decision version of the minimum feedback vertex set problem, namely, deciding whether there exists an FVS of size at most k, has been proved to be NP-complete [25]. It is actually among the earliest problems for which NP-completeness has been shown. Finding the minimum FVS for general graphs is still an active research area.

The fastest algorithm to date has the complexity of $\mathcal{O}(1.7548^n)$, where n is the number of nodes [14].

However, if we have some prior knowledge about the graph structure, optimal or near optimal solutions can be found efficiently or even in linear time for many special graph structures. In [35] the authors showed that in reducible graphs the minimum FVS can be found in linear time. In [41] and [26] polynomial time algorithms are proposed to find the minimum FVS for cyclically reducible graphs and AT-free graphs. Fixed parameter polynomial time algorithms are also developed to find the minimum FVS if the minimum size is known to be bounded by a parameter [9].

In addition, approximate algorithms are also proposed to give an FVS whose size is bounded by a factor times the minimum possible size [1, 2, 12].

In [1] the authors propose a factor 2 approximate algorithm that gives an FVS of size at most 2 times the minimum size. We introduce this algorithm in this section. First we describe some useful notions and operations related to this approximate minimum FVS selection algorithm, such as clean graphs, graph cleaning and semi-disjoint cycles. More detailed descriptions can be found in [1].

A graph is called a *clean graph* if there are no leaf nodes, i.e. nodes of degree one or zero. The *graph cleaning* operation is to remove the leaf nodes with their incident edges and keep removing the newly generated leaf nodes until the remaining graph is a clean graph. In other words, the graph cleaning operation removes all the tree branches in a graph. For a graph with n nodes, the graph cleaning operation can be done in time $\mathcal{O}(n)$, assuming the degree of each node is already available.

A *disjoint* cycle is a cycle in a graph where all nodes in it have degree 2. A *semi-disjoint cycle* is a cycle where all nodes in the cycle have degrees 2 with at most one exception. For instance, the cycle *abcda* in Figure 2-3 is a semi-disjoint cycle with node a being the exception. A path with only degree-2 nodes is called an *non-branching path.* Detecting semi-disjoint cycles can be done in time $\mathcal{O}(n+m)$.

We summarize the factor 2 minimum FVS selection algorithm in Figure 2-4.

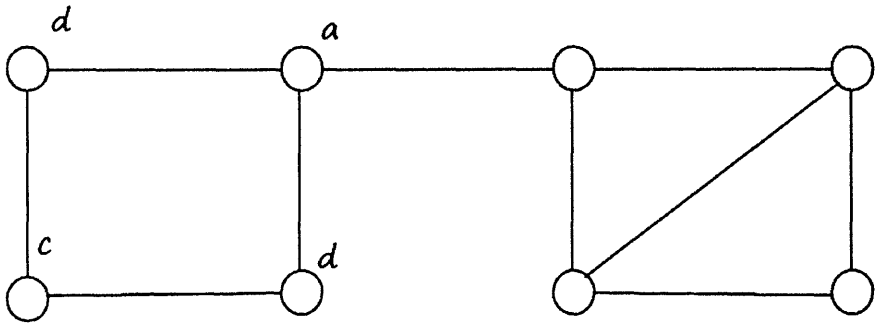


Figure 2-3: A semi-disjoint cycle *abcda*

Input: an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: a feedback vertex set $\mathcal{F}$

1. Initialize $\mathcal{F} = \emptyset$ and $w(i) = 1, \ \forall i \in \mathcal{V}$.
2. Clean up $\mathcal{G}$. Update $\mathcal{V}$ to be the set of the remaining nodes.
3. While $\mathcal{V} \neq \emptyset$, repeat
   - If $\mathcal{G}$ contains a semi-disjoint cycle $\mathcal{C}$, then
     $\gamma = \min\{w(i); i \in \mathcal{V}\}$ and let $w(i) \leftarrow w(i) - \gamma, \ \forall i \in \mathcal{C}$.
     Else
     $\gamma = \min\{w(i)/(\deg(i) - 1); i \in \mathcal{V}\}$ and let $w(i) \leftarrow w(i) - \gamma(\deg(u) - 1), \ \forall i \in \mathcal{V}$.
   - Remove each node i with $w(i) = 0$ from $\mathcal{V}$ and add it to $\mathcal{F}$. Push it onto stack S.
   - Clean up $\mathcal{G}$.
4. While stack $S \neq \emptyset$, repeat
   - $i = \text{pop}(S)$
   - If $\mathcal{F} - \{i\}$ is an FVS in the original $\mathcal{G}$, remove i from $\mathcal{F}$.

Figure 2-4: A factor 2 approximate algorithm for the minimum FVS problem

The complexity of this algorithm is given as follows.

Proposition 2.7. *The algorithm in Figure 2-4 gives a minimal FVS of size at most two times the size of the minimum FVS in time* $\mathcal{O}(\min\{m \log n, n^2\})$*, where* m *and* n *are the numbers of the edges and vertices respectively.*

An FVS is called a *minimal FVS* if it does not contain any smaller FVS. In other words, if we remove any node from a minimal FVS, it is no longer an FVS.

Proof. See [1]. □

Chapter 3

Exact Feedback Message Passing

As we discussed in the background chapter, inference on a tree-structured graph can be solved exactly by BP in linear time with respect to the graph size. For graphs with cycles, the convergence and correctness of LBP are not guaranteed. In this chapter, we show that for a class of graphs with small feedback vertex set (FVS), inference problems can be solved efficiently by the feedback message passing (FMP) algorithm.

The high level idea of the FMP algorithm is to select a set of nodes as feedback nodes. We use a special message update scheme for the feedback nodes while using standard BP message passing scheme for the non-feedback nodes. The feedback nodes are selected to break all the cycles in the graph. There are two rounds of message passing in FMP, each requires some amount of memory and communication capacity, according to the graph structure. In the FMP algorithm, we perform two rounds of BP with different parameters. In the first round of BP, we obtain the inaccurate "partial variance" and "partial mean" as well as some "feedback gain" for every non-feedback node. Next we compute the exact inference results for the feedback nodes. In the second round of BP, we make corrections to the "partial variance" and "partial mean" of the non-feedback nodes.

In this chapter we start with the single feedback node case, and illustrate the algorithm by concrete examples. Then we describe the general FMP algorithm with multiple feedback nodes. We will also discuss the correctness and complexity of the FMP algorithm.

3.1 The Single Feedback Node Case

3.1.1 FMP with a Single Feedback Node

Consider the loopy graph in Figure 3-1(a). A Gaussian graphical model is defined on this graph. Let J and $\mathbf{h}$ be the information matrix and potential vector of the model respectively. In this graph every cycle passes through node 1, which can be used as the single feedback node for the graph. We denote the set of neighboring nodes of node 1 as $\mathcal{N}(1)$. We use $\mathcal{T}$ to denote the subgraph excluding node 1 and its incident edges. Graph $\mathcal{T}$ is a tree or a disconnected forest, which does not have any cycles. Here we call such graphs *tree-structured* graphs. With node 1 being the feedback node, the FMP algorithm has the following steps:

Step 1: Initialization

We construct a new potential vector $\mathbf{h}^1$ on $\mathcal{T}$, where $h_i^1 = J_{1i}, \forall i \in \mathcal{N}(1)$ and $h_i^1 = 0, \forall i \notin \mathcal{N}(1), i \neq 1$. We can view this step as node 1 sending messages to its neighbors to initiate $\mathbf{h}^1$. The new potential vector $\mathbf{h}^1$ captures some of node 1's effects on $\mathcal{T}$ so that nodes in $\mathcal{T}$ can process this information. See Figure 3-1(b) for illustration.

Step 2: First Round of BP

BP is performed on $\mathcal{T}$ with model parameters $J_\mathcal{T}$ and $\mathbf{h}_\mathcal{T}$, where $J_\mathcal{T}$ and $\mathbf{h}_\mathcal{T}$ are the corresponding submatrix and subvector of J and $\mathbf{h}$ respectively. After convergence each node i in $\mathcal{T}$ obtains its "partial variance" $P_{ii}^\mathcal{T} = (J_\mathcal{T}^{-1})_{ii}$ and its "partial mean" $\mu_i^\mathcal{T} = (J_\mathcal{T}^{-1}\mathbf{h}_\mathcal{T})_i$ by the standard BP on $\mathcal{T}$. Note that these results are not the true variances and means since they only capture local structures within $\mathcal{T}$ without considering the effects of node 1. Another BP is performed on $\mathcal{T}$ with the the same information matrix $J_\mathcal{T}$ and the new potential vector $\mathbf{h}^1$. Each node i on $\mathcal{T}$ will get a *feedback gain* g_i^1, where $g_i^1 = (J_\mathcal{T}^{-1}\mathbf{h}^1)_i$ by standard BP on $\mathcal{T}$.

In practice we only need to run BP once, with $J_\mathcal{T}$ and two potential vectors $\mathbf{h}_\mathcal{T}$ and $\mathbf{h}^1$ simultaneously. We also put the header information "1" into the messages related to $\mathbf{h}^1$ to mark the source of the messages. Since $\mathcal{T}$ is a tree-structured graph, BP converges after a finite number of iterations in linear time. After convergence, every node i knows its "partial

variance" $P_{ii}^{\mathcal{T}} = (J_{\mathcal{T}}^{-1})_{ii}$, "partial mean" $\boldsymbol{\mu}_i^{\mathcal{T}} = (J_{\mathcal{T}}^{-1}\mathbf{h}^{\mathcal{T}})_i$ and feedback gain $g_i^1 = (J_{\mathcal{T}}^{-1}\mathbf{h}^1)_i$.

Step 3: Inference for the Feedback Node

Feedback node 1 collects the feedback gains from its neighbors as shown in Figure 3-1(d). Node 1 then calculates the *exact* variance and mean for itself by

$$P_{11} = (J_{11} - \sum_{j \in \mathcal{N}(1)} J_{1j} g_j^1)^{-1}, \tag{3.1}$$

$$\mu_1 = P_{11}(h_1 - \sum_{j \in \mathcal{N}(1)} J_{1j} \mu_j^{\mathcal{T}}). \tag{3.2}$$

In this step, all the computations only involve local parameters and the feedback gain or "partial mean" of node 1's neighbors.

Step 4: Feedback Message Passing

After feedback node 1 obtains its own variance and mean, it passes the results to other nodes in order to correct their "partial variances" $P_{ii}^{\mathcal{T}}$ and "partial means" $\mu_i^{\mathcal{T}}$ as computed in Step 2 (see Figure 3-1(e)).

The neighbors of node 1 revise their node potentials as follows:

$$\tilde{h}_j = \begin{cases} h_j - J_{1j}\mu_1, & \forall j \in \mathcal{N}(1) \\ h_j, & \forall j \notin \mathcal{N}(1). \end{cases} \tag{3.3}$$

From this formula we see that only node 1's neighbors revise their node potentials. The revised potential vector $\tilde{\mathbf{h}}_{\mathcal{T}}$ is then be used in the second round of BP.

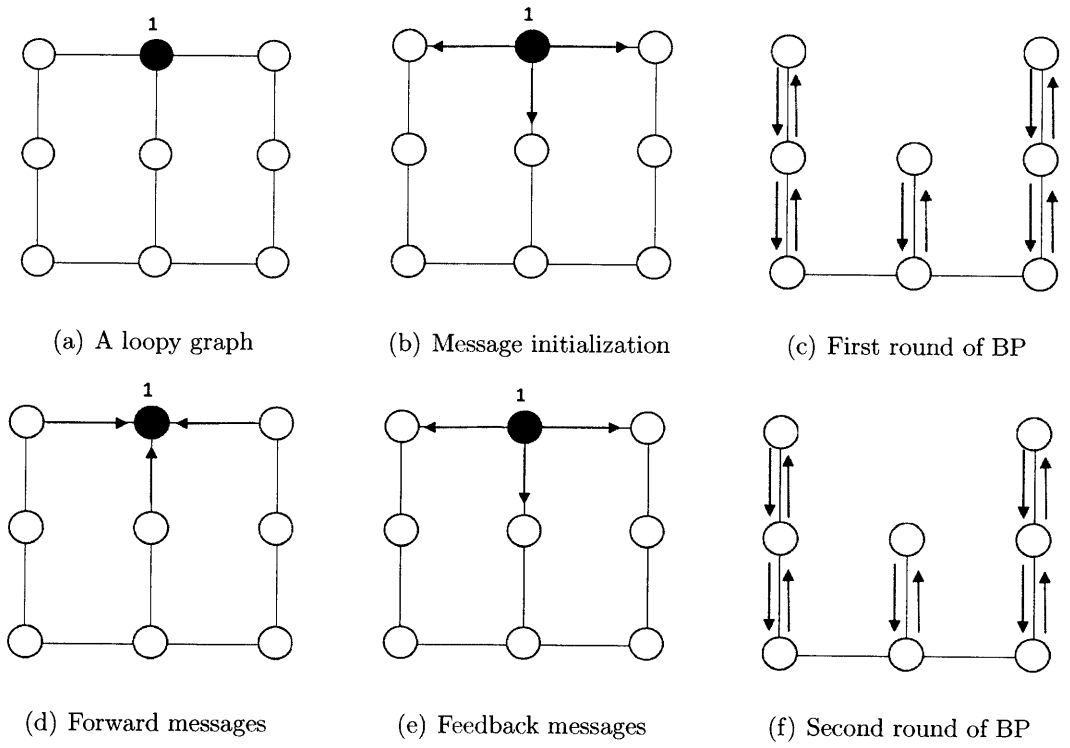
Step 5: Second Round of BP

We perform BP on $\mathcal{T}$ with $J_{\mathcal{T}}$ and $\tilde{\mathbf{h}}_{\mathcal{T}}$ (see Figure 3-1(f)). The means $\mu_i = (J_{\mathcal{T}}^{-1}\tilde{\mathbf{h}}_{\mathcal{T}})_i$ obtained from BP are the *exact* means.

The *exact* variances can be computed by adding correction terms to the "partial variances".

$$P_{ii} = P_{ii}^{\mathcal{T}} + P_{11}(g_i^1)^2, \forall i \in \mathcal{T}, \tag{3.4}$$

where the "partial variance" $P_{ii}^{\mathcal{T}}$ and the feedback gain g_i^1 are computed in Step 2. This computation is local when P_{11} is propagated throughout the graph as a message.



(a) A loopy graph (b) Message initialization (c) First round of BP

(d) Forward messages (e) Feedback messages (f) Second round of BP

Figure 3-1: The FMP algorithm with a single feedback node

In Appendix A, we will give another example of the FMP algorithm with a single node. We write down all the feedback messages explicitly.

3.1.2 A Walk-sum Analysis for FMP with a Single Feedback Node

We now give a walk-sum interpretation of why the FMP algorithm with a single feedback node gives the exact inference results. A more rigorous proof is provided when we discuss the FMP algorithm for general graphs.

As we described in the background chapter, the exact means and variances in Gaussian graphical models can be interpreted as sums of walks on the graphs. If we can show that the solution given by an algorithm captures exactly all walks needed, the correctness of the algorithm is then verified. In this section, we will show that the FMP algorithm with a single feedback node indeed captures exactly all necessary walks for correct inference.

Assume there exists a single feedback node in the graph $\mathcal{G}$ of n nodes. We number the

feedback node as node 1 and we denote the tree-structured graph achieved by eliminating node 1 as $\mathcal{T}$. A Gaussian graphical model with information matrix J and potential vector $\mathbf{h}$ is Markov with respect to $\mathcal{G}$. Without loss of generality, we can assume J has been normalized so that all the entries in the diagonal equal to one. Let $R = I - J$, where I is the identity matrix. Following the notations in the background chapter, $\forall i \neq j, R_{ij} = -J_{ij}$ is the weight of a single step from node i to node j. By (2.22), the variance of node 1 can be calculated by

$$P_{11} = \phi(1 \xrightarrow{\mathcal{G}} 1), \tag{3.5}$$

where $\phi(1 \xrightarrow{\mathcal{G}} 1)$ denotes the sum of all walks starting from and ending at node 1 within $\mathcal{G}$. We call $\phi(1 \xrightarrow{\mathcal{G}} 1)$ the *self-return walk-sum* of node 1.

By (2.23), we have

$$\phi(1 \xrightarrow{\mathcal{G}} 1) = \frac{1}{1 - \phi^{(1)}(1 \xrightarrow{\mathcal{G}} 1)}, \tag{3.6}$$

where $\phi^{(1)}(1 \xrightarrow{\mathcal{G}} 1)$ is called the *single-visit self-return walk-sum* of node 1. It is defined as the sum of all the walks from node 1 to node 1 within $\mathcal{G}$ that do not visit node 1 in between.

For any such single-visit self-return walk, it must visit a neighbor j at the first step from node 1 and visit a neighbor k at the last step before it ends at node 1. In between, the walk can be any walk from j to k within $\mathcal{T}$. This decomposition guarantees that the walk visit node 1 exactly once after it leaves node 1. By enumerating the combinations of j and k, we can compute $\phi^{(1)}(1 \xrightarrow{\mathcal{G}} 1)$ by

$$\phi^{(1)}(1 \xrightarrow{\mathcal{G}} 1) = \sum_{j \in \mathcal{N}(1)} \sum_{k \in \mathcal{N}(1)} \phi(j \xrightarrow{\mathcal{T}} k) J_{1j} J_{k1}. \tag{3.7}$$

We can further simply the expression as

$$\begin{aligned} \phi^{(1)}(1 \xrightarrow{\mathcal{G}} 1) &= \sum_{k \in \mathcal{N}(1)} J_{1k} \Big(\sum_{j \in \mathcal{N}(1)} J_{1j} \phi(j \xrightarrow{\mathcal{T}} k) \Big) && (3.8) \\ &= \sum_{k \in \mathcal{N}(1)} J_{1k} g_k^1, && (3.9) \end{aligned}$$

where $g_k^1 = \sum_{j \in \mathcal{N}(1)} J_{1j}\phi(j \xrightarrow{\mathcal{T}} k)$.

By (2.22), the mean of node i is a weighted sum of walk-sums:

$$\mu_i = \sum_{j \in \mathcal{G}} h_j \phi(j \xrightarrow{\mathcal{G}} i). \tag{3.10}$$

Comparing the expression of g_k^1 with (3.10), g_k^1 can be interpreted as the "mean" of node k when we consider a model defined on $\mathcal{T}$ with the submatrix $J_{\mathcal{T}}$ of J as the information matrix and $\mathbf{h}^1 = J_{\mathcal{T},1}$ as the potential vector. By Proposition 2.1, g_k^1 is calculated exactly by performing BP, since $\mathcal{T}$ is a tree-structured graph. Therefore, g_k^1 here is the same as g_k^1 in (3.1).

By substituting (3.9) into (3.6), we have

$$P_{11} = (1 - \sum_{k \in \mathcal{N}(1)} J_{1k} g_k^1)^{-1}. \tag{3.11}$$

Since J_{11} has been normalized to 1, the formula is the same as in the FMP algorithm. Therefore, the variance of the feedback node 1 by the FMP algorithm is exact.

By (2.24),

$$\phi(i \xrightarrow{\mathcal{G}} 1) = \phi^{(1)}(i \xrightarrow{\mathcal{G}} 1)\phi(1 \xrightarrow{\mathcal{G}} 1), \forall i \neq 1, \tag{3.12}$$

where $\phi^{(1)}(i \xrightarrow{\mathcal{G}} 1)$ is the single-visit walk-sum from i to 1. It is defined as the sum of all the walks from i to 1 that visit node 1 exactly once.

The interpretation of this walk-sum equation is that any walk from i to 1 can be uniquely decomposed into two parts: the walk starting from i until its first visit to node 1; any walk starting from 1 and ending at 1. We can further categorize a walk in the first part according to which node it visits before its first visit to 1. Then

$$\phi^{(1)}(i \xrightarrow{\mathcal{G}} 1) = \sum_{j \in \mathcal{N}(1)} \phi(i \xrightarrow{\mathcal{T}} j)(-J_{j1}) = -g_k^1, \tag{3.13}$$

where by our notation, $-J_{j1} = R_{j1}$ is the weight of a single step walk from j to 1.

With the above formulas, we can show that the mean of the feedback node 1 is computed

exactly by the FMP algorithm. By (3.10),

$$\mu_1 = \sum_i h_i \phi(i \xrightarrow{\mathcal{G}} 1) \tag{3.14}$$

$$= \sum_{i \neq 1} h_i \phi(i \xrightarrow{\mathcal{G}} 1) + h_1 \phi(1 \xrightarrow{\mathcal{G}} 1). \tag{3.15}$$

By (3.12) and (3.13),

$$\mu_1 = \sum_{i \neq 1} h_i \phi^{(1)}(i \xrightarrow{\mathcal{G}} 1)\phi(1 \xrightarrow{\mathcal{G}} 1) + h_1 \phi(1 \xrightarrow{\mathcal{G}} 1) \tag{3.16}$$

$$= (\sum_{i \neq 1} h_i \sum_{j \in \mathcal{N}(1)} \phi(i \xrightarrow{\mathcal{T}} j)(-J_{j1}))\phi(1 \xrightarrow{\mathcal{G}} 1) + h_1 \phi(1 \xrightarrow{\mathcal{G}} 1) \tag{3.17}$$

$$= (- \sum_{j \in \mathcal{N}(1)} J_{j1} \sum_{i \neq 1} h_i \phi(i \xrightarrow{\mathcal{T}} j) + h_1)\phi(1 \xrightarrow{\mathcal{G}} 1) \tag{3.18}$$

$$= (h_1 - \sum_{j \in \mathcal{N}(1)} J_{j1} \mu_j^{\mathcal{T}}) P_{11}, \tag{3.19}$$

where $\mu_j^{\mathcal{T}} = \sum_{i \neq 1} h_i \phi(i \xrightarrow{\mathcal{T}} j)$. Here $\mu_j^{\mathcal{T}}$ can be exactly computed by running BP in $\mathcal{T}$ with $J_{\mathcal{T}}$ and $h_{\mathcal{T}}$. It is the same $\mu_j^{\mathcal{T}}$ as in the FMP algorithm. This shows that the FMP algorithm gives the exact mean for the feedback node.

We continue to show that FMP also gives the correct means and variances for nodes in $\mathcal{T}$ after the second round of BP.

$$\text{For } i \neq 1, \quad \mu_i = \sum_j h_j \phi(j \xrightarrow{\mathcal{G}} i) \tag{3.20}$$

$$= \sum_{j \neq 1} h_j \phi(j \xrightarrow{\mathcal{G}} i) + h_1 \phi(1 \xrightarrow{\mathcal{G}} i). \tag{3.21}$$

In this formula, $\phi(j \xrightarrow{\mathcal{G}} i)$ is the sum of all the walks in $\mathcal{G}$ from j to i. These walks can be classified to two categories: all walks from j to i within $\mathcal{T}$ and all the walks that visit 1 at least once. We denote the sum of the latter walks as $\phi_1^{(1)}(i \xrightarrow{\mathcal{G}} j)$. Then

$$\phi(j \xrightarrow{\mathcal{G}} i) = \phi(j \xrightarrow{\mathcal{T}} i) + \phi_1^{(1)}(i \xrightarrow{\mathcal{G}} j). \tag{3.22}$$

Note that we will frequently use the fact that walk-sum is symmetric, namely, $\phi(i \xrightarrow{\mathcal{G}} j) = \phi(j \xrightarrow{\mathcal{G}} i)$ in our following steps. It is because the weight of any walk remains the same when we reverse the order of every step. Any walk that visits node 1 at least once can be uniquely decomposed into: a walk within $\mathcal{T}$ from i to one of 1's neighbors; a step from 1's neighbor k to node 1; and any walk from 1 to j within $\mathcal{G}$. We have

$$\phi_1^{(1)}(i \xrightarrow{\mathcal{G}} j) = \sum_{k \in \mathcal{N}(1)} \phi(i \xrightarrow{\mathcal{T}} k)(-J_{k1})\phi(1 \xrightarrow{\mathcal{G}} j). \tag{3.23}$$

Substituting (3.22) and (3.23) into (3.21), we have

$$\begin{aligned} \mu_i &= \sum_{j \neq 1} h_j [\phi(j \xrightarrow{\mathcal{T}} i) + \sum_{k \in \mathcal{N}(1)} \phi(i \xrightarrow{\mathcal{T}} k)(-J_{k1})\phi(1 \xrightarrow{\mathcal{G}} j)] + h_1 \phi(1 \xrightarrow{\mathcal{G}} i) && (3.24) \\ &= \sum_{j \neq 1} h_j \phi(j \xrightarrow{\mathcal{T}} i) + \sum_{j \neq 1} h_j \sum_{k \neq 1} \left[(-J_{1k})\phi(i \xrightarrow{\mathcal{T}} k)\phi(1 \xrightarrow{\mathcal{G}} j)\right] + h_1 \phi(1 \xrightarrow{\mathcal{G}} i). && (3.25) \end{aligned}$$

By (3.12) and (3.13),

$$\begin{aligned} \mu_i &= \sum_{j \neq 1} h_j \phi(j \xrightarrow{\mathcal{T}} i) + \sum_{j} h_j \sum_{k \neq 1} (-J_{1k})\phi(i \xrightarrow{\mathcal{T}} k)\phi(1 \xrightarrow{\mathcal{G}} j) && (3.26) \\ &= \sum_{j \neq 1} h_j \phi(j \xrightarrow{\mathcal{T}} i) + \sum_{k} h_k \sum_{j \neq 1} (-J_{1j})\phi(i \xrightarrow{\mathcal{T}} j)\phi(1 \xrightarrow{\mathcal{G}} k) && (3.27) \\ &= \sum_{j \neq 1} h_j \phi(j \xrightarrow{\mathcal{T}} i) + \sum_{j \neq 1} (-J_{1j})[\sum_{k} h_k \phi(1 \xrightarrow{\mathcal{G}} k)]\phi(i \xrightarrow{\mathcal{T}} j). && (3.28) \end{aligned}$$

Since $\mu_1 = \sum_k h_k \phi(1 \xrightarrow{\mathcal{G}} k)$,

$$\begin{aligned} \mu_i &= \sum_{j \neq 1} h_j \phi(j \xrightarrow{\mathcal{T}} i) + \sum_{j \neq 1} (-J_{1j})\mu_1 \phi(i \xrightarrow{\mathcal{T}} j) && (3.29) \\ &= \sum_{j \neq 1} (h_j - J_{1j}\mu_1)\phi(j \xrightarrow{\mathcal{T}} i) && (3.30) \\ &= \sum_{j \neq 1} \widetilde{h}_j \phi(j \xrightarrow{\mathcal{T}} i), && (3.31) \end{aligned}$$

where $\widetilde{h}_j = h_j - J_{1j}\mu_1$. It is the same as the revised node potential $\widetilde{h}_j$ in the FMP algorithm. The last equation is equivalent to performing BP on $\mathcal{T}$ with the revised the potential vector

$\widetilde{\mathbf{h}}_{\mathcal{T}}$ as in the FMP algorithm. This shows the correctness of the means in the FMP algorithm.

Finally, we show that the variances of nodes in $\mathcal{T}$ are also computed exactly in the FMP algorithm. The exact variance of node i in $\mathcal{T}$ is the sum of all the walks starting from and ending at node i. These walks can be categorized into all the walks within $\mathcal{T}$ and all the walks that visit node 1 at least once. Therefore,

$$\phi(i \xrightarrow{\mathcal{G}} i) = \phi(i \xrightarrow{\mathcal{T}} i) + \phi_1^{(1)}(i \xrightarrow{\mathcal{G}} j). \tag{3.32}$$

Any such walk can be decomposed into three parts: a single visit walk from i to 1; any walk from i to i; and the reverse of a single-visit walk from i to 1. Therefore,

$$\phi(i \xrightarrow{\mathcal{G}} i) = \phi(i \xrightarrow{\mathcal{T}} i) + \phi^{(1)}(i \xrightarrow{\mathcal{G}} 1)\phi(i \xrightarrow{\mathcal{G}} i)\phi^{(1)}(i \xrightarrow{\mathcal{G}} 1). \tag{3.33}$$

By (3.13),

$$P_{ii} = P_{ii}^{\mathcal{T}} + (-g_i^1)P_{11}(-g_i^1) = P_{ii}^{\mathcal{T}} + P_{11}(g_i^1)^2, \tag{3.34}$$

where $P_{ii}^{\mathcal{T}}$, g_i^1 and P_{11} are all the same quantities as in the FMP algorithm. This completes our walk-sum analysis for the FMP algorithm with a single feedback node.

3.2 Feedback Message Passing for General Graphs

For a general graph, the removal of a single node may not break all the cycles. We may need multiple nodes to form an FVS. The FMP algorithm with a single feedback node can be extended by adding extra feedback messages. Each extra message corresponds to one feedback node in the FVS. Since the computational complexity depends on the size of the FVS, it is important to find a small FVS before we apply the FMP algorithm.

There are many algorithms to find a small FVS. We can use an exact algorithm to find the minimum FVS or we can use an efficient approximate algorithm to find an FVS of small size, but not necessarily the minimum size. For example, the algorithm described in Figure 2-4 gives an FVS of size at most two times the minimum size in $\mathcal{O}(\min\{m \log n, n^2\})$, where m is the number of edges and n is the number of nodes. This algorithm is efficient and has been widely used. If we want to solve many problems with different parameters but the same structure, we may instead use a more complex algorithm to obtain a smaller FVS. How to choose the appropriate FVS selection algorithm depends on practical problems. In this section, we assume the FVS $\mathcal{F}$ is given by certain algorithm before we perform the FMP algorithm.

3.2.1 Feedback Message Passing Algorithm

Without loss of generality, we order the nodes such that nodes in $\mathcal{F}$ are the first k nodes, where k is the size of $\mathcal{F}$. The FMP algorithm with multiple feedback nodes is essentially the same as with a single feedback node. Now k extra potential vectors are used instead of just one. In the single feedback node case, we can directly compute the mean and variance for the feedback node after the first round of BP. Here we need to solve an inference problem on a smaller graph of size k. The exact variances are computed by adding multiple correction terms to the "partial variances". The exact means are computed by a second round of BP with a revised potential vector, which has "listened" to the feedback nodes.

The FMP algorithm with a given FVS $\mathcal{F}$ is summarized in Figure 3-2.

Input: information matrix J, potential vector $\mathbf{h}$ and feedback vertex set $\mathcal{F}$ of size k
Output: mean μ_i and variance P_{ii} for every node i

1. Construct k extra potential vectors: $\forall p \in \mathcal{F}, \mathbf{h}^p = J_{\mathcal{T},p}$, each corresponding to one feedback node.

2. Perform BP on $\mathcal{T}$ with $J_{\mathcal{T}}$, $h_{\mathcal{T}}$ to obtain $P_{ii}^{\mathcal{T}} = (J_{\mathcal{T}}^{-1})_{ii}$ and $\mu_i^{\mathcal{T}} = (J_{\mathcal{T}}^{-1}\mathbf{h}_{\mathcal{T}})_i$ for each $i \in \mathcal{T}$. With the k extra potential vectors, calculate the feedback gains $g_i^1 = (J_{\mathcal{T}}^{-1}\mathbf{h}^1)_i, g_i^2 = (J_{\mathcal{T}}^{-1}\mathbf{h}^2)_i, \ldots, g_i^k = (J_{\mathcal{T}}^{-1}\mathbf{h}^k)_i$ for $i \in \mathcal{T}$ by BP .

3. Obtain a size k subgraph with $\widehat{J}_{\mathcal{F}}$ and $\widehat{\mathbf{h}}_{\mathcal{F}}$ given by

$$\begin{aligned}(\widehat{J}_{\mathcal{F}})_{pq} &= J_{pq} - \sum_{j \in \mathcal{N}(p) \cap \mathcal{T}} J_{pj} g_j^q, \ \forall p, q \in \mathcal{F} \\ (\widehat{\mathbf{h}}_{\mathcal{F}})_p &= h_p - \sum_{j \in \mathcal{N}(p) \cap \mathcal{T}} J_{pj} \mu_j^{\mathcal{T}}, \ \forall p \in \mathcal{F},\end{aligned}$$

   and solve the inference problem on the small graph by $P^{\mathcal{F}} = \widehat{J}_{\mathcal{F}}^{-1}$, $\boldsymbol{\mu}^{\mathcal{F}} = \widehat{J}_{\mathcal{F}}^{-1}\widehat{\mathbf{h}}_{\mathcal{F}}$.

4. Revise the potential vector on $\mathcal{T}$ by

$$\tilde{h}_i = h_i - \sum_{j \in \mathcal{N}(i) \cap \mathcal{F}} J_{ij} \boldsymbol{\mu}_j^{\mathcal{F}}, \ \forall i \in \mathcal{T}.$$

5. Another round of BP with the revised potential vector $\tilde{\mathbf{h}}_{\mathcal{T}}$ gives the exact means for nodes on $\mathcal{T}$.

   Add correction terms to obtain the exact variances for nodes in $\mathcal{T}$:

$$P_{ii} = P_{ii}^{\mathcal{T}} + \sum_{p \in \mathcal{F}} \sum_{q \in \mathcal{F}} g_i^p P_{pq}^{\mathcal{F}} g_i^q, \ \forall i \in \mathcal{T}.$$

Figure 3-2: The Feedback Message Passing algorithm

3.2.2 Correctness and Complexity of the FMP algorithm

In this section, we analyze the correctness and computational complexity of the FMP algorithm.

Theorem 3.1. *The feedback message passing algorithm described in Figure 3-2 gives the exact means and exact variances for all nodes.*

Proof. Given the model parameters J and $\mathbf{h}$, with an FVS $\mathcal{F}$ of size k, we want to compute the variances of all nodes, which are the diagonal entries of $P = J^{-1}$, and compute the means $\boldsymbol{\mu} = J^{-1}\mathbf{h}$ of all nodes. Without loss of generality, we order the nodes such that the feedback nodes are the first k nodes. The other nodes are numbered from $k+1$ to n. Following this ordering, we can write the model parameters as

$$J = \begin{bmatrix} J_{\mathcal{F}} & J'_M \\ J_M & J_{\mathcal{T}} \end{bmatrix} \text{ and } \mathbf{h} = \begin{bmatrix} \mathbf{h}_{\mathcal{F}} \\ \mathbf{h}_{\mathcal{T}} \end{bmatrix}. \tag{3.35}$$

For a valid model, $J \succ 0$. Then we have $J_{\mathcal{F}} \succ 0$, $J_{\mathcal{T}} \succ 0$ and $J_{\mathcal{F}} - J'_M J_{\mathcal{T}}^{-1} J_M \succ 0$. Note that $J_{\mathcal{T}}$ is tree-structured by our definition of $\mathcal{F}$.

Similarly, we can write

$$P = \begin{bmatrix} P_{\mathcal{F}} & P'_M \\ P_M & P_{\mathcal{T}} \end{bmatrix} \text{ and } \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{F}} \\ \boldsymbol{\mu}_{\mathcal{T}} \end{bmatrix}. \tag{3.36}$$

By the above notations,

$$J_M = [\mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^k], \tag{3.37}$$

where $\mathbf{h}^1$, $\mathbf{h}^2$,..., $\mathbf{h}^k$ are the same as in the FMP algorithm. The feedback gain $\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^k$ in the FMP algorithm are computed by BP with $\mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^k$ as the potential vectors.

By Proposition 2.1, BP gives the exact inference results on trees. Therefore,

$$[\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^k] = \left[J_{\mathcal{T}}^{-1}\mathbf{h}^1, J_{\mathcal{T}}^{-1}\mathbf{h}^2, \ldots, J_{\mathcal{T}}^{-1}\mathbf{h}^k\right] = J_{\mathcal{T}}^{-1} J_M. \tag{3.38}$$

In the FMP algorithm, $\boldsymbol{\mu}^{\mathcal{T}}$ is computed by BP using $h_{\mathcal{T}}$ as the potential vector, then

$$\boldsymbol{\mu}^{\mathcal{T}} = J_{\mathcal{T}}^{-1}\mathbf{h}_{\mathcal{T}}. \tag{3.39}$$

In the FMP algorithm, the diagonal of $J_{\mathcal{T}}^{-1}$ is also calculated exactly in the first round of BP as $P_{ii}^{\mathcal{T}} = (J_{\mathcal{T}}^{-1})_{ii}$.

Since $P = J^{-1}$, we have

$$\begin{bmatrix} J_{\mathcal{F}} & J_M' \\ J_M & J_{\mathcal{T}} \end{bmatrix} \begin{bmatrix} P_{\mathcal{F}} & P_M' \\ P_M & P_{\mathcal{T}} \end{bmatrix} = \begin{bmatrix} I_{\mathcal{F}} & \mathbf{0} \\ \mathbf{0} & I_{\mathcal{T}} \end{bmatrix}. \tag{3.40}$$

Then

$$J_M P_M' + J_{\mathcal{T}} P_{\mathcal{T}} = I_{\mathcal{T}} \tag{3.41}$$

$$J_M P_{\mathcal{F}} + J_{\mathcal{T}} P_M = \mathbf{0}. \tag{3.42}$$

Thus

$$P_{\mathcal{T}} = J_{\mathcal{T}}^{-1} - J_{\mathcal{T}}^{-1} J_M P_M' \tag{3.43}$$

$$P_M' = -P_{\mathcal{F}}(J_{\mathcal{T}}^{-1} J_M)'. \tag{3.44}$$

By substituting (3.44) to (3.43), we have

$$P_{\mathcal{T}} = J_{\mathcal{T}}^{-1} + (J_{\mathcal{T}}^{-1} J_M) P_{\mathcal{F}} (J_{\mathcal{T}}^{-1} J_M)'. \tag{3.45}$$

By (3.38), we have

$$P_{\mathcal{T}} = J_{\mathcal{T}}^{-1} + [\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^k] P_{\mathcal{F}} [\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^k]'. \tag{3.46}$$

Therefore,

$$P_{ii} = P_{ii}^{\mathcal{T}} + \sum_{p\in\mathcal{F}}\sum_{q\in\mathcal{F}} \mathbf{g}_i^p \left(P_{\mathcal{F}}\right)_{pq} \mathbf{g}_i^q, \quad \forall i \in \mathcal{T}, \tag{3.47}$$

where $\mathbf{g}^p$ is computed in the FMP algorithm. Here $P_{\mathcal{F}}$ is the exact covariance matrix of the feedback nodes in $\mathcal{F}$. This is the same equation as in the FMP algorithm. To show that the FMP algorithm gives the correct variances, later we will show that $P_{\mathcal{F}}$ is indeed calculated exactly in the FMP algorithm.

Since $\boldsymbol{\mu} = J^{-1}\mathbf{h}$, we have

$$\begin{bmatrix} J_{\mathcal{F}} & J'_M \\ J_M & J_{\mathcal{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{F}} \\ \boldsymbol{\mu}_{\mathcal{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_{\mathcal{F}} \\ \mathbf{h}_{\mathcal{T}} \end{bmatrix}. \tag{3.48}$$

Then

$$J_M\boldsymbol{\mu}_{\mathcal{F}} + J_{\mathcal{T}}\boldsymbol{\mu}_{\mathcal{T}} = \mathbf{h}_{\mathcal{T}} \tag{3.49}$$

$$\boldsymbol{\mu}_{\mathcal{T}} = J^{-1}(\mathbf{h}_{\mathcal{T}} - J_M\boldsymbol{\mu}_{\mathcal{F}}). \tag{3.50}$$

We define $\widetilde{\mathbf{h}}_{\mathcal{T}} = \mathbf{h}_{\mathcal{T}} - J_M\boldsymbol{\mu}_{\mathcal{F}}$, namely

$$(\widetilde{\mathbf{h}}_{\mathcal{T}})_i = \begin{cases} \mathbf{h}_i - \sum_{j\in\mathcal{N}(i)\cap\mathcal{F}} J_{ij}\mu_j & i \in \mathcal{N}(\mathcal{F}) \\ \mathbf{h}_i & i \notin \mathcal{N}(\mathcal{F}), \end{cases} \tag{3.51}$$

where $\mathcal{N}(\mathcal{F}) = \bigcup_{j\in\mathcal{F}}\mathcal{N}(j)\backslash\mathcal{F}$, and $\boldsymbol{\mu}_{\mathcal{F}}$ is the exact mean of nodes in $\mathcal{F}$. This step is equivalent to performing BP with parameters $J_{\mathcal{T}}$ and the revised potential vector $\widetilde{\mathbf{h}}_{\mathcal{T}}$ as in the FMP algorithm.

To complete the proof, we need to show that the FMP algorithm indeed gives the exact $P_{\mathcal{F}}$ and $\boldsymbol{\mu}_{\mathcal{F}}$.

By the equation of Schur's complement,

$$\widehat{J}_{\mathcal{F}} \triangleq P_{\mathcal{F}}^{-1} = J_{\mathcal{F}} - J'_M J_{\mathcal{T}}^{-1} J_M \succ 0 \tag{3.52}$$

$$\widehat{h}_{\mathcal{F}} \triangleq P_{\mathcal{F}}^{-1}\boldsymbol{\mu}_{\mathcal{F}} = \mathbf{h}_{\mathcal{F}} - J'_M J_{\mathcal{T}}^{-1}\mathbf{h}_{\mathcal{T}}. \tag{3.53}$$

Therefore,

$$\widehat{J}_{\mathcal{F}} = J_{\mathcal{F}} - J'_M[\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^k] \tag{3.54}$$

$$\widehat{h}_{\mathcal{F}} = \mathbf{h}_{\mathcal{F}} - J'_M \boldsymbol{\mu}^{\mathcal{T}}. \tag{3.55}$$

This is exactly the same as in the FMP algorithm:

$$(\widehat{J}_{\mathcal{F}})_{pq} = J_{pq} - \sum_{j \in \mathcal{N}(p) \cap \mathcal{T}} J_{pj} g_j^q, \ \forall p, q \in \mathcal{F}$$

$$(\widehat{\mathbf{h}}_{\mathcal{F}})_p = h_p - \sum_{j \in \mathcal{N}(p) \cap \mathcal{T}} J_{pj} \boldsymbol{\mu}_j^{\mathcal{T}}, \ \forall p \in \mathcal{F}.$$

Therefore, we get the exact mean and exact covariance matrix for nodes in $\mathcal{F}$ by solving $P_{\mathcal{F}} = (\widehat{J}_{\mathcal{F}})^{-1}$ and $\boldsymbol{\mu}_{\mathcal{F}} = P_{\mathcal{F}} \widehat{\mathbf{h}}_{\mathcal{F}}$. This completes the proof. □

Proposition 3.2. *The computational complexity of the FMP algorithm in Figure 3-2 is $\mathcal{O}(k^2 n)$, where k is the size of the FVS and n is the total number of nodes in the graph.*

Proof. We analyze the complexity step by step (see Figure 3-2). In step 1 and step 2, BP is performed on $\mathcal{T}$ with $k+2$ messages. The total complexity is $\mathcal{O}(k(n-k))$. In step 3, $\mathcal{O}(k^2(n-k))$ computations are needed to obtain $\widehat{J}_{\mathcal{F}}$ and $\widehat{h}_{\mathcal{F}}$. It takes $\mathcal{O}(k^3)$ to solve the inference problem on a graph of size k. Step 4 takes $\mathcal{O}(k(n-k))$ computations to give the exact means and step 5 takes $\mathcal{O}(k^2(n-k))$ computations to add correction terms. Therefore, the total complexity is $\mathcal{O}(k^2 n)$. □

According to the above results, if the size of the FVS is bounded by a constant, the means and variances can be computed exactly in linear time. If the size of the FVS is unbounded but grows much slower than the graph size, e.g. the size of the FVS is $\mathcal{O}(\log n)$, the algorithm is still very efficient. For example, based on a hierarchical tree model, we can add an additional node to each layer, and connect it to every other nodes in the same layer. In this way, the dependence structure within the same layer can be explicitly modeled. Figure 3-3 shows an example of such models with 4 layers. For such a model of n nodes,

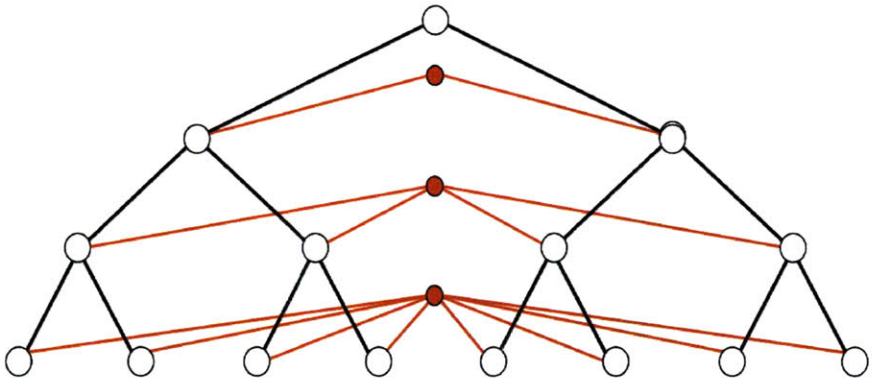


Figure 3-3: A hierarchical model with an additional node in each layer

the number of layers is $\mathcal{O}(\log n)$. The set of additional nodes can be used as an FVS in the FMP algorithm. The complexity of exact inference on such models is then $\mathcal{O}(n(\log n)^2)$.

Chapter 4

Approximate Feedback Message Passing

Though the exact FMP algorithm always gives correct inference results, it can be intractable when the size of the FVS is large. If the size of the FVS k grows linearly with the graph size n, the computational complexity of the algorithm will be as large as $\mathcal{O}(n^3)$, which is the same order as solving a linear system by Gaussian elimination. In this case, we can make some approximations to obtain tractable algorithms.

The bottlenecks of the exact FMP algorithm are solving a size k inference problem and adding k^2 correction terms to each of the non-feedback node. One natural way to make an approximation is to use a set of feedback nodes of bounded size. Then we do calculations among the non-feedback nodes approximately while solving the size k inference problem exactly. The resulting algorithm is called the *approximate FMP algorithm with a pseudo-FVS*.

In this chapter we describe this approximate algorithm and relevant theoretical results.

4.1 Approximate FMP with a Pseudo-FVS

An FVS breaks all cycles in a graph. In contrast, we define a *pseudo-FVS* as a subset of an FVS that does not break all the cycles. When the size of the FVS is large, the exact FMP algorithm becomes intractable. Then we can replace the FVS by a pseudo-FVS and make approximations accordingly. A useful pseudo-FVS has a small size but breaks most cycles in the graph or most "crucial" cycles in terms of resulting inference error. In this section, we introduce how to extend the exact FMP algorithm to the approximate FMP algorithm with a given pseudo-FVS.

Consider a Gaussian graphical model with underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We use $\widetilde{\mathcal{F}}$ to denote a pseudo-FVS, and use $\widetilde{\mathcal{T}}$ to denote the *pseudo-tree* obtained by eliminating nodes in $\widetilde{\mathcal{F}}$. Being consistent with the exact FMP algorithm, we call nodes in $\widetilde{\mathcal{F}}$ the feedback nodes.

A natural extension is to replace BP by LBP in Step 2 and Step 5 of the exact FMP algorithm (see Figure 3-2) since $\widetilde{\mathcal{T}}$ still has cycles. However, the convergence and correctness of message passing in $\widetilde{\mathcal{T}}$ is no longer guaranteed when an arbitrary pseudo-FVS is chosen. Obtaining convergence and obtaining accurate results are the goals of selecting a good pseudo-FVS, a topic which is discussed later. Compared to LBP on the entire graph, the approximate FMP with a pseudo-FVS yields better convergence and accuracy while the complexity scales favorably with the graph size. Another advantage of the algorithm is that we can explicitly trade off the accuracy and efficiency by controlling the size of the pseudo-FVS.

The approximate FMP algorithm with a given pseudo-FVS of size k is summarized in Figure 4-1. Without loss of generality, we order the nodes such that nodes in $\widetilde{\mathcal{F}}$ are the first k nodes. The total complexity of this algorithm in Figure 4-1 depends on the graph size, the cardinality of the pseudo-FVS, and the number of iterations of LBP within the pseudo-tree, as described in the following proposition.

Proposition 4.1. *The total computational complexity of the approximate FMP algorithm with a pseudo-FVS in Figure 4-1 is $\mathcal{O}(km(k+D))$, where k is the size of the pseudo-FVS, m is the total number of edges in the graph, and D is the number of iterations in each round*

Input: information matrix J, potential vector $\mathbf{h}$, and pseudo-FVS $\widetilde{\mathcal{F}}$ of size k
Output: mean μ_i and variance P_{ii} of every node i

1. Construct k extra potential vectors: $\mathbf{h}^p = J_{\widetilde{\mathcal{T}},p}, \forall p \in \widetilde{\mathcal{F}}$, each corresponding to one feedback node.

2. Perform LBP on $\widetilde{\mathcal{T}}$ with $J_{\widetilde{\mathcal{T}}}$, $h_{\widetilde{\mathcal{T}}}$ and the k extra potential vectors. After convergence, obtain $P_{ii}^{\widetilde{\mathcal{T}}}$, $\mu_i^{\widetilde{\mathcal{T}}}$ and k feedback gains $g_i^1, g_i^2, \ldots, g_i^k$ for each node i in $\widetilde{\mathcal{T}}$.

3. Obtain a size-k subgraph with $\widehat{J}_{\widetilde{\mathcal{F}}}$ and $\widehat{\mathbf{h}}_{\widetilde{\mathcal{F}}}$ given by

$$\begin{aligned}(\widehat{J}_{\widetilde{\mathcal{F}}})_{pq} &= J_{pq} - \sum_{j \in \mathcal{N}(p) \cap \widetilde{\mathcal{T}}} J_{pj} g_j^q, \ \forall p, q \in \widetilde{\mathcal{F}} \\ (\widehat{\mathbf{h}}_{\widetilde{\mathcal{F}}})_p &= h_p - \sum_{j \in \mathcal{N}(p) \cap \widetilde{\mathcal{T}}} J_{pj} \mu_j^{\widetilde{\mathcal{T}}}, \ \forall p \in \widetilde{\mathcal{F}},\end{aligned}$$

   and solve the inference problem on the small graph by $P^{\widetilde{\mathcal{F}}} = \widehat{J}_{\widetilde{\mathcal{F}}}^{-1}$, $\boldsymbol{\mu}^{\widetilde{\mathcal{F}}} = \widehat{J}_{\widetilde{\mathcal{F}}}^{-1} \widehat{\mathbf{h}}_{\widetilde{\mathcal{F}}}$.

4. Revise the potential vector on $\widetilde{\mathcal{T}}$ by

$$\tilde{h}_i = h_i - \sum_{j \in \mathcal{N}(i) \cap \widetilde{\mathcal{F}}} J_{ij} \mu_j^{\widetilde{\mathcal{F}}}, \ \forall i \in \widetilde{\mathcal{T}}.$$

5. Perform another round of LBP with the revised potential vector $\tilde{\mathbf{h}}_{\widetilde{\mathcal{T}}}$ to obtain the means for nodes in $\widetilde{\mathcal{T}}$.

   Add correction terms to obtain the variances of nodes in $\widetilde{\mathcal{T}}$:

$$P_{ii} = P_{ii}^{\widetilde{\mathcal{T}}} + \sum_{p \in \widetilde{\mathcal{F}}} \sum_{q \in \widetilde{\mathcal{F}}} g_i^p P_{pq}^{\widetilde{\mathcal{F}}} g_i^q, \ \forall i \in \widetilde{\mathcal{T}}.$$

Figure 4-1: The approximate FMP algorithm with a pseudo-FVS

of LBP within the pseudo-tree.

Proof. In step 1 and step 2, LBP is performed on $\widetilde{\mathcal{T}}$ with $k+2$ messages. The complexity is $\mathcal{O}(kmD)$. In step 3, $\mathcal{O}(k^2(n-k))$ computations are needed to obtain $\widehat{J}_{\mathcal{F}}$ and $\widehat{h}_{\mathcal{F}}$, where n is the number of nodes in the graph. It takes $\mathcal{O}(k^3)$ computations to solve the inference problem on the size-k subgraph. In step 4 and step 5, it takes $\mathcal{O}(mD)$ computations to obtain the means by another round of LBP, and $\mathcal{O}(k^2(n-k))$ computations to add correction terms. The total complexity is thus $\mathcal{O}(km(k+D))$. □

Note that this complexity result is for general graphs, which can be very dense graphs or even complete graphs. If the graphs we consider are sparse, e.g. the number of edges grows linearly with the number of nodes, the computational complexity is then $\mathcal{O}(km(k+D))$.

4.2 Convergence and Accuracy

In this section, we provide some theoretical results on the convergence and accuracy of the approximate FMP algorithm with a pseudo-FVS.

For convenience, we first describe the following lemma which is used in our proofs of Theorem 4.3.

Lemma 4.2. *For any nonnegative matrix A, $\rho(A_T) \leq \rho(A)$, where A_T is a principle submatrix of A with column and row indices T, and $\rho(\cdot)$ denotes the spectral radius of a matrix.*

Proof. See the proof of corollary 8.1.20 in [19]. □

Theorem 4.3. *Consider a Gaussian graphical model with underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and model parameters J and $\mathbf{h}$. If the model is walk-summable, the approximate FMP algorithm converges with any selection of pseudo-FVS $\widetilde{\mathcal{F}} \subset \mathcal{V}$.*

Proof. First we normalize J such that all the diagonal entries equal to one:

$$\bar{J} = D^{-1/2} J D^{-1/2}, \tag{4.1}$$

where D is a diagonal matrix with the same diagonal as J.

Let $R = I - \bar{J}$ and $(\bar{R})_{ij} = |R_{ij}|$. Then $\bar{R}$ has zero diagonal and nonnegative entries. We call $\bar{R}$ the *absolute edge weight matrix.*

By Proposition 2.3, a graphical model is walk-summable if and only if $\rho(\bar{R}) < 1$, where $\rho(\bar{R})$ is the spectral radius of the absolute edge weight matrix.

In the approximate FMP algorithm, LBP is performed on the pseudo-tree induced by $\widetilde{\mathcal{T}} = \mathcal{V} \backslash \widetilde{\mathcal{F}}$. The information matrix on the pseudo-tree is $J_{\widetilde{\mathcal{T}}}$, which is a submatrix of J. We define $\bar{R}^{\widetilde{\mathcal{T}}}$ as the absolute edge weight matrix of the model on $\widetilde{\mathcal{T}}$. By simple algebra, we can see $\bar{R}^{\widetilde{\mathcal{T}}}$ is the same as $\bar{R}_{\widetilde{\mathcal{T}}}$, the submatrix of $\bar{R}$ corresponding to columns and rows $\widetilde{\mathcal{T}}$.

By Lemma 4.2,

$$\rho(\bar{R}^{\mathcal{T}}) = \rho(\bar{R}_{\widetilde{\mathcal{T}}}) \leq \rho(\bar{R}) < 1 \tag{4.2}$$

By Proposition 2.5, LBP on $\widetilde{\mathcal{T}}$ is guaranteed to converge. Except for LBP on $\widetilde{\mathcal{T}}$, all the computations have finite steps for any choice of $\widetilde{\mathcal{F}}$. Therefore, the approximate FMP algorithm converges with any $\widetilde{\mathcal{F}} \subset \mathcal{V}$. □

In practice, the approximate FMP algorithm often converges with a proper selection of pseudo-FVS even if LBP on the entire graph does not converge.

Theorem 4.4. *Consider a Gaussian graphical model with parameters J and $\mathbf{h}$. If the approximate FMP algorithm in Figure 4-1 converges with a pseudo-FVS $\widetilde{\mathcal{F}}$, it gives the correct means and variances for all nodes in $\widetilde{\mathcal{F}}$.*

Proof. Without loss of generality, we order the nodes such that nodes in $\widetilde{\mathcal{F}}$ are the first k nodes, where k is the size of the pseudo-FVS. Then we have

$$J = \begin{bmatrix} J_{\widetilde{\mathcal{F}}} & J'_M \\ J_M & J_{\widetilde{\mathcal{T}}} \end{bmatrix} \text{ and } \mathbf{h} = \begin{bmatrix} \mathbf{h}_{\widetilde{\mathcal{F}}} \\ \mathbf{h}_{\widetilde{\mathcal{T}}} \end{bmatrix}. \tag{4.3}$$

By the formula of Schur's complement,

$$\begin{aligned}\widehat{J}_{\widetilde{\mathcal{F}}} &\triangleq P_{\widetilde{\mathcal{F}}}^{-1} = J_{\widetilde{\mathcal{F}}} - J'_M J_{\widetilde{\mathcal{T}}}^{-1} J_M \succ 0 \\ \widehat{h}_{\widetilde{\mathcal{F}}} &\triangleq P_{\widetilde{\mathcal{F}}}^{-1} \boldsymbol{\mu}_{\widetilde{\mathcal{F}}} = \mathbf{h}_{\widetilde{\mathcal{F}}} - J'_M J_{\widetilde{\mathcal{T}}}^{-1} \mathbf{h}_{\widetilde{\mathcal{T}}},\end{aligned} \tag{4.4}$$

where $P_{\widetilde{\mathcal{F}}}$ and $\boldsymbol{\mu}_{\widetilde{\mathcal{F}}}$ are the exact covariance matrix and mean vector of nodes in $\widetilde{\mathcal{F}}$.

By Proposition 2.2, when LBP converges, it gives the correct means. Applying this theorem on graph $\widetilde{\mathcal{T}}$, we have

$$\begin{aligned}\mathbf{g}^i &= J_{\widetilde{\mathcal{T}}}^{-1} J_{\widetilde{\mathcal{T}},i}, \ \forall i = 1, 2, \ldots, k \\ \boldsymbol{\mu}^{\widetilde{\mathcal{T}}} &= J_{\widetilde{\mathcal{T}}}^{-1} \mathbf{h}_{\widetilde{\mathcal{T}}},\end{aligned} \tag{4.5}$$

where $\mathbf{g}^i$ is the feedback gain and $\boldsymbol{\mu}^{\widetilde{\mathcal{T}}}$ is the partial mean as described in Figure 4-1. These quantities are exact after convergence.

Substituting (4.5) into (4.4), we have

$$\begin{aligned}\widehat{J}_{\widetilde{\mathcal{F}}} &= J_{\widetilde{\mathcal{F}}} - J'_M [\mathbf{g}^1, \mathbf{g}^2, \ldots, \mathbf{g}^k] \\ \widehat{h}_{\widetilde{\mathcal{F}}} &= \mathbf{h}_{\widetilde{\mathcal{F}}} - J'_M \boldsymbol{\mu}^{\widetilde{\mathcal{T}}}.\end{aligned} \tag{4.6}$$

This is equivalent to

$$\begin{aligned}(\widehat{J}_{\widetilde{\mathcal{F}}})_{pq} &= J_{pq} - \sum_{j \in \mathcal{N}(p) \cap \widetilde{\mathcal{T}}} J_{pj} g_j^q, \ \forall p, q \in \widetilde{\mathcal{F}} \\ (\widehat{\mathbf{h}}_{\widetilde{\mathcal{F}}})_p &= h_p - \sum_{j \in \mathcal{N}(p) \cap \widetilde{\mathcal{T}}} J_{pj} \boldsymbol{\mu}_j^{\widetilde{\mathcal{T}}}, \ \forall p \in \widetilde{\mathcal{F}},\end{aligned} \tag{4.7}$$

which is the same formula as in Step 3 in Figure 4-1. By solving the small inference problem on $\widetilde{\mathcal{F}}$ in Step 3, we obtain the exact means and variances for all nodes in $\widetilde{\mathcal{F}}$. □

Theorem 4.5. *Consider a Gaussian graphical model defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with model parameters J and $\mathbf{h}$. Assume the approximate FMP algorithm in Figure 4-1 converges*

with a pseudo-FVS $\widetilde{\mathcal{F}}$. Then the calculated mean of any node in $\widetilde{\mathcal{T}} = \mathcal{V}\backslash\widetilde{\mathcal{F}}$ is correct; the calculated variance of node i in $\widetilde{\mathcal{T}}$ equals the sum of all the backtracking walks of node i within $\widetilde{\mathcal{T}}$ plus all the self-return walks of node i that visit at least one node in $\widetilde{\mathcal{F}}$.

Proof. Without loss of generality, we assume the information matrix J has been normalized to have unit diagonal. We order the nodes such that nodes in $\widetilde{\mathcal{F}}$ are the first k nodes, where k is the size of $\widetilde{\mathcal{F}}$. We have

$$J = \begin{bmatrix} J_{\widetilde{\mathcal{F}}} & J'_M \\ J_M & J_{\widetilde{\mathcal{T}}} \end{bmatrix} \text{ and } \mathbf{h} = \begin{bmatrix} \mathbf{h}_{\widetilde{\mathcal{F}}} \\ \mathbf{h}_{\widetilde{\mathcal{T}}} \end{bmatrix}. \tag{4.8}$$

Following the same steps as in the proof of Theorem 3.1, we have

$$\boldsymbol{\mu}_{\widetilde{\mathcal{T}}} = J_{\widetilde{\mathcal{T}}}^{-1}(\mathbf{h}_{\widetilde{\mathcal{T}}} - J_M \boldsymbol{\mu}_{\widetilde{\mathcal{F}}}). \tag{4.9}$$

By Proposition 4.4, $\boldsymbol{\mu}_{\widetilde{\mathcal{F}}}$ is computed exactly in Step 3 in Figure 4-1. By Theorem 2.2, LBP on $\widetilde{\mathcal{T}}$ gives the exact result of $J_{\widetilde{\mathcal{T}}}^{-1}\mathbf{h}^*$ for any potential vector $\mathbf{h}^*$. Regarding $\mathbf{h}_{\widetilde{\mathcal{T}}} - J_M\boldsymbol{\mu}_{\widetilde{\mathcal{F}}}$ as $\mathbf{h}^*$, we can obtain exact $\boldsymbol{\mu}_{\widetilde{\mathcal{T}}}$ after another round of LBP in Step 5. Therefore, the means of all nodes in $\widetilde{\mathcal{T}}$ are exact.

Similarly, following the steps in the proof of Theorem 3.1, we have

$$P_{\widetilde{\mathcal{T}}} = J_{\widetilde{\mathcal{T}}}^{-1} + (J_{\widetilde{\mathcal{T}}}^{-1}J_M)P_{\widetilde{\mathcal{F}}}(J_{\widetilde{\mathcal{T}}}^{-1}J_M)'. \tag{4.10}$$

The exact variance of node $i \in \widetilde{\mathcal{T}}$ equals the sum of all the self-return walks of node i according to (2.22). We categorize these walks into two classes: self-return walks of node i within $\widetilde{\mathcal{T}}$, and self-returns walks that visit at least one node in $\widetilde{\mathcal{F}}$.

The diagonal of $J_{\widetilde{\mathcal{T}}}^{-1}$ captures exactly the first class of walks. Then the second term in the right-hand side of (4.10) corresponds to the sums of the second class of walks. Since $J_{\widetilde{\mathcal{T}}}^{-1}J_M$ and $P_{\widetilde{\mathcal{F}}}$ are obtained exactly by the algorithm according to the proof of Theorem 4.4, the sums of the second class of walks are exact. We thus only need to study the accuracy of the approximation to the first term computed by our algorithm.

By Proposition 2.6, LBP on $\widetilde{\mathcal{T}}$ gives the sums of all the backtracking walks after convergence. Therefore, the calculated variance result of node i in $\widetilde{\mathcal{T}}$ equals to the sum of all the backtracking walks of node i within $\widetilde{\mathcal{T}}$ plus all the self-return walks of node i that visit at least one node in $\widetilde{\mathcal{F}}$. □

From the previous two theorems, we can see that the approximate FMP algorithm with a pseudo-FVS always gives the exact means for all nodes after convergence. It captures more walks than LBP in calculating the variances: for nodes in $\widetilde{\mathcal{F}}$, it captures all self-return walks while LBP only captures the backtracking walks; for nodes in $\widetilde{\mathcal{T}}$, in addition to the backtracking walks it also captures the non-backtracking self-return walks that visit $\widetilde{\mathcal{F}}$ while LBP does not captures any such walks.

Corollary 4.6. *Assume a graphical model is walk-summable. $\widetilde{\mathcal{F}}_1, \widetilde{\mathcal{F}}_2, \ldots, \widetilde{\mathcal{F}}_k$ are pseudo-FVS satisfying $\emptyset \neq \widetilde{\mathcal{F}}_1 \subseteq \widetilde{\mathcal{F}}_2 \subseteq \cdots \subseteq \widetilde{\mathcal{F}}_k$ and $\mathcal{F} \supseteq \widetilde{\mathcal{F}}_k$ is an FVS. Then $W_i^{LBP} \subseteq W_i^{\widetilde{\mathcal{F}}_1} \subseteq W_i^{\widetilde{\mathcal{F}}_2} \subseteq \ldots \subseteq W_i^{\widetilde{\mathcal{F}}_k} \subseteq W_i^{\mathcal{F}}$ for any node i in the graph. Here W_i^{LBP} is the set of walks captured by LBP for calculating the variance of node i; $W_i^{\widetilde{\mathcal{F}}_j}$ is the set of walks captured by the approximate FMP algorithm with pseudo-FVS $\widetilde{\mathcal{F}}_j$; and $W_i^{\mathcal{F}}$ is the set of walks captured by the exact FMP algorithm with FVS $\mathcal{F}$.*

Proof. All of the algorithms here converge since the model is walk-summable. When we increase the set of feedback nodes, we miss fewer walks needed for the exact variances. □

Theorem 4.7. *Consider an attractive Gaussian graphical model with graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Let $\widetilde{\mathcal{F}}_1 \subseteq \widetilde{\mathcal{F}}_2 \subseteq \cdots \subseteq \widetilde{\mathcal{F}}_k \subseteq \mathcal{F}$ denote the pseudo-FVS (FVS), and P_i^{LBP}, $P_i^{\widetilde{\mathcal{F}}_1}$, ..., $P_i^{\widetilde{\mathcal{F}}_k}$, $P_i^{\mathcal{F}}$ denote the corresponding variances calculated for node i by LBP, approximate FMP and the exact FMP algorithm. P_i is the exact variance of node i. We have $P_i^{LBP} \leq P_i^{\widetilde{\mathcal{F}}_1} \leq P_i^{\widetilde{\mathcal{F}}_2} \leq \cdots \leq P_i^{\widetilde{\mathcal{F}}_k} \leq P_i^{\mathcal{F}} = P_i$ for any node i in $\mathcal{V}$.*

Proof. By Proposition 2.4, a valid Gaussian graphical model is walk-summable if it is attractive. Then all the algorithms mentioned in the theorem are guaranteed to converge.

By Corollary 4.6, $W_i^{LBP} \subseteq W_i^{\widetilde{\mathcal{F}}_1} \subseteq W_i^{\widetilde{\mathcal{F}}_2} \subseteq \ldots \subseteq W_i^{\widetilde{\mathcal{F}}^k} \subseteq W_i^{\mathcal{F}}$ for any node i, where we follow the same notations as in Corollary 4.6. Since all walks have nonnegative weight in an

attractive model, the sum of walks of a smaller set is less than or equal to the sum of walks of a larger set.

Therefore,

$$P_i^{LBP} \leq P_i^{\widetilde{\mathcal{F}}^1} \leq P_i^{\widetilde{\mathcal{F}}^2} \leq \cdots \leq P_i^{\widetilde{\mathcal{F}}^k} \leq P_i^{\mathcal{F}} = P_i, \ \forall i \in \mathcal{V}. \tag{4.11}$$

□

4.3 Finding a Pseudo-FVS of Bounded size

As we have discussed in previous chapters, the problems of divergence and inaccuracy of LBP are due to the existence of cycles. By removing a set of feedback nodes from the graph, we can reduce the number of cycles. One goal of choosing a pseudo-FVS is to ensure that LBP converges on the remaining pseudo-tree; the other goal is to obtain smaller errors. Breaking all the cycles by using an FVS will achieve both goals at the same time since the convergence and correctness of the exact FMP algorithm are both guaranteed. However, it may lead to intractable algorithms when the FVS is large. This motivates us to find a good pseudo-FVS whose removal breaks most cycles or most *crucial cycles* in the graph. In addition, by adjusting the size of the pseudo-FVS, we can explicitly trade off the accuracy and efficiency. In this section we discuss two algorithms motivated by these two goals.

4.3.1 Selecting a Pseudo-FVS for Convergence Purpose

Consider a Gaussian graphical model of n nodes with information matrix J and potential vector $\mathbf{h}$. Without loss of generality, we assume J has been normalized to have unit diagonal.

Let $\bar{R}$ denote the matrix taken the entrywise absolute values of $R = I - J$. By Proposition 2.5, $\rho(\bar{R}_{\mathcal{G}}) < 1$ is a sufficient condition for LBP to converge on graph $\mathcal{G}$, where $\rho(\cdot)$ denotes the spectral radius of a matrix. Hence the problem reduces to that of removing the minimum number of nodes such that $\rho(\bar{R}_{\widetilde{\mathcal{T}}}) < 1$ for the remaining graph $\widetilde{\mathcal{T}}$. This problem can be written as

$$
\begin{aligned}
\text{minimize} \quad & k \\
\text{subject to} \quad & (1-\epsilon)I - X\bar{R}X \succeq \mathbf{0} \\
& (1-\epsilon)I + X\bar{R}X \succeq \mathbf{0} \\
& \sum_i X_{ii} = n - k \\
& X \text{ is an } n \times n \text{ diagonal matrix, } X_{ii} \in \{0,1\},
\end{aligned}
\tag{4.12}
$$

where ϵ is a small positive number.

Alternatively, we can bound the number of nodes to remove and minimize $\rho(\bar{R}_{\widetilde{\mathcal{T}}})$ for the remaining graph $\widetilde{\mathcal{T}}$. This problem can then be written as

$$
\begin{aligned}
\text{minimize} \quad & \tau \\
\text{subject to} \quad & \tau I - X\bar{R}X \succeq \mathbf{0} \\
& \tau I + X\bar{R}X \succeq \mathbf{0} \\
& \sum_i X_{ii} \geq n - k \\
& X \text{ is a } n \times n \text{ diagonal matrix, } X_{ii} \in \{0,1\}.
\end{aligned}
\tag{4.13}
$$

Since directly solving this problem is intractable, we make several relaxations to obtain a simple algorithm.

In [19] the following bound on the spectral radius of a nonnegative matrix is given:

$$
\min_i \sum_j \bar{R}_{ij} \leq \rho(\bar{R}) \leq \max_i \sum_j \bar{R}_{ij}.
$$

Instead of minimizing the spectral radius, we minimize its upper bound. Then we have the following problem formulation:

Input: information matrix J and the maximum size k of the pseudo-FVS
Output: a pseudo-FVS $\widetilde{\mathcal{F}}$

1. Let $\widetilde{\mathcal{F}} = \emptyset$ and normalize J to have unit diagonal.
2. Repeat until $|\widetilde{\mathcal{F}}| = k$ or the remaining graph is empty.
   (a) Clean up the graph by eliminating all the tree branches.
   (b) Update the scores $s(i) = \sum_{j \in \mathcal{N}(i)} |J_{ij}|$.
   (c) Put the node with the largest score into $\widetilde{\mathcal{F}}$ and remove it from the current graph.

Figure 4-2: Algorithm 1 for selecting a pseudo-FVS

$$\begin{aligned} \text{minimize} \quad & \tau \\ \text{subject to} \quad & \mathbf{1}^T X \bar{R} X \leq \tau \mathbf{1} \\ & \sum_i X_{ii} \geq n - k \\ & X \text{ is a } n \times n \text{ diagonal matrix, } X_{ii} \in \{0, 1\}. \end{aligned}$$

We further simplify this problem by a greedy heuristic: we select one feedback node at each iteration. Node i with the largest score $s(i) = \sum_j \bar{R}_{ij}$ is selected at each iteration. Then we remove node i from the graph and put it into $\widetilde{\mathcal{F}}$. We continue the same procedure on the remaining graph until the maximum allowed size k of $\widetilde{\mathcal{F}}$ is reached or the remaining graph does not have any cycles. The algorithm is summarized in Figure 4-2.

4.3.2 Selecting a Pseudo-FVS for Accuracy Purpose

Consider a Gaussian graphical model with n nodes, information matrix J and potential vector $\mathbf{h}$. Without loss of generality, we assume J has been normalized to have unit diagonal. We want to find a pseudo-FVS $\widetilde{\mathcal{F}}$ of bounded size k to give the smallest error. Let $R = I - J$ be the edge weight matrix. R_{ij} is the weight of a single-step walk from i to j or from j to i. We define the error as the sum of the absolute errors of the means and variances for all

nodes. Namely,

$$\epsilon = \sum_{i \in \mathcal{V}} |\widehat{P}_{ii} - P_{ii}| + \sum_{i \in \mathcal{V}} |\widehat{\mu}_i - \mu_i|,$$

where $\widehat{P}_{ii}$ and $\widehat{\mu}_i$ are the variances and means computed by some algorithms; P_{ii} and μ_i are the exact variances and means.

By Proposition 2.2 and Theorem 4.4, 4.5, the means by LBP and by the approximate FMP are both correct for all nodes after convergence. Hence we only need to consider the errors of the variances.

From a walk-sum perspective,

$$\epsilon(LBP) = \sum_{i \in \mathcal{V}} |\phi(i \xrightarrow{\text{NB}} i)|, \tag{4.14}$$

where $\epsilon(LBP)$ is the error given by LBP and $\phi(i \xrightarrow{\text{NB}} i)$ denotes the sum of all non-backtracking self-return walks of node i.

Let $\epsilon(p)$ denote the error given by the approximate FMP with a single feedback node p. At each iteration, we choose a single node p with smallest error $\epsilon(p)$, namely,

$$p^* = \operatorname*{argmin}_{p \in \mathcal{V}} \epsilon(p). \tag{4.15}$$

This is equivalent to

$$p^* = \operatorname*{argmax}_{p \in \mathcal{V}} \epsilon(LBP) - \epsilon(p), \tag{4.16}$$

since $\epsilon(LBP)$ does not depend on the choice of p.

By Theorem 4.5, when we use node p as the single feedback node, the variance of p is correct while the variances of other nodes lack the non-backtracking self-return walks within $\mathcal{V} \backslash p$. Then

$$\epsilon(p) = \sum_{i \in \mathcal{V} \backslash p} |\phi(i \xrightarrow{\text{NB in } \mathcal{V} \backslash p} i)|, \tag{4.17}$$

where $\phi(i \xrightarrow{\text{NB in } \mathcal{V} \backslash p} i)$ denotes the sum of non-backtracking self-return walks of node i within

$\mathcal{V}\backslash p$. Then

$$\epsilon(LBP) - \epsilon(p) = |\phi(p \xrightarrow{NB} p)| + \sum_{i \in \mathcal{V}\backslash p} (|\phi(i \xrightarrow{\text{NB}} i)| - |\phi(i \xrightarrow{\text{NB in } \mathcal{V}\backslash p} i)|). \tag{4.18}$$

Until now, we have not made any approximations. Maximizing the above function of p gives the best p to choose as a single feedback node at each iteration. However, the exact function is difficult to deal with, and we must make approximations. For simplicity, we assume the model is attractive so that the weight of all walks are nonnegative. Then

$$\epsilon(LBP) - \epsilon(p) = \phi(p \xrightarrow{\text{NB}} p) + \sum_{i \in \mathcal{V}\backslash p} \phi(i \xrightarrow{\text{NB visit } p} i), \tag{4.19}$$

where $\phi(i \xrightarrow{\text{NB visit } p} i)$ denotes the sum of all the non-backtracking self-return walks of node i that visit p at least once.

All walks involved in the above formula are non-backtracking self-return walks and they all visit p at least once. A complete set of such walks is difficult to capture. Instead, we can consider a subset of such walks, i.e. all cycles passing through p. Then we replace the left hand side of (4.19) by $\sum_{C \in \mathcal{C}_p} w(C)l(C)$, where $\mathcal{C}_p$ is the set of all cycles passing through node p, $w(C)$ is the weight of a cycle C, and $l(C)$ is the length of a cycle C.

Enumerating all the cycles is sometimes intractable since it involves global structure of the graph. We approximate the lengths and weights of the cycles by some simplifications: we use the average length l to replace $l(C)$; the weight of a cycle passing through node p and its neighbors j and q can be approximated by $r^{l-2}R_{jp}R_{qp}$, where r is the average weight of the edges; if we clean up all the tree branches (see Chapter 2 for graph cleaning) before we select a feedback node, we can assume there exists a cycle passing through any pair of p's neighbors. Therefore, the objective function becomes $\sum_{j,q \in \mathcal{N}(p), j<q} r^{l-2}R_{jp}R_{qp}l$. Thus we

Input: information matrix J and the maximum size k of the pseudo-FVS
Output: a pseudo-FVS $\widetilde{\mathcal{F}}$

1. Let $\widetilde{\mathcal{F}} = \emptyset$ and normalize J to have unit diagonal.
2. Repeat until $|\widetilde{\mathcal{F}}| = k$ or the remaining graph is empty.
   (a) Clean up the graph by eliminating all the tree branches.
   (b) Update the scores $s(i) = \sum_{l,k \in \mathcal{N}(i), l<k} |J_{il} J_{ik}|$.
   (c) Put the node with the largest score into $\widetilde{\mathcal{F}}$ and remove it from the current graph.

Figure 4-3: Algorithm 2 for selecting a pseudo-FVS

can formulate an alternative optimization problem as

$$\tilde{p}^* = \underset{p \in \mathcal{V}}{\operatorname{argmax}} \sum_{j,q \in \mathcal{N}(p), j<q} r^{l-2} R_{jp} R_{qp} l \tag{4.20}$$

$$= \underset{p \in \mathcal{V}}{\operatorname{argmax}} \sum_{j,q \in \mathcal{N}(p), j<q} |J_{jp} J_{qp}|. \tag{4.21}$$

The algorithm is summarized in Figure 4-3.

4.3.3 Comments

The two algorithms are motivated from convergence and accuracy purposes respectively. Both of the algorithms tend to choose nodes with large edge weights as feedback nodes. Actually, the score function of Algorithm 1 is equivalent to $(\sum_{j \in \mathcal{N}(i)} |J_{ij}|)^2 = \sum_{j,q \in \mathcal{N}(i)} |J_{ij} J_{iq}|$, which is very similar to the score function of $s(i) = \sum_{j,q \in \mathcal{N}(i), j<q} |J_{ij} J_{iq}|$ in Algorithm 2. This similarity is consistent with the fact that when $\widetilde{\mathcal{F}}$ becomes an FVS, the two goals—convergence and accuracy are achieved at the same time.

The complexity of the two algorithms are both $\mathcal{O}(km)$, where m is the number of edges and k is the size of the pseudo FVS. These two algorithms are very fast in practice since they only involve simple computations.

An adaptive way to select a pseudo-FVS is also favorable: we can first use the criterion

for convergence and then switch to the criterion for accuracy after convergence is guaranteed.

Finding a proper pseudo-FVS is important to obtain convergent and accurate inference algorithms. We will see later in Chapter 5 that there is a huge performance difference between a good selection and a bad selection of $\tilde{\mathcal{F}}$.

4.4 Regional FMP Algorithm

When the size of the FVS $\mathcal{F}$ of a Gaussian graphical model is large, we can use the approximate FMP algorithm with a pseudo FVS, in which an inference problem within the feedback nodes is solved exactly while LBP in the pseudo-tree gives approximate results. There is an alternative way to make approximations: we still use the full FVS $\mathcal{F}$, and perform BP exactly in the tree part while solving the inference problem on the FVS approximately.

When we use a full FVS of large size k, there are two bottlenecks in the computational complexity. First, solving the inference problem on the feedback nodes is difficult because of the large size. Second, we need to add k^2 correction terms to each non-feedback node. For the first problem, we can use a sparse "thinned" graph instead of a complete graph of the feedback nodes. For the second problem, we can only add a small number of correction terms to each non-feedback node. There are many ways to construct a "thinned" graph and to choose correction terms. Here we propose the *regional FMP algorithm*, which chooses the "thinned" graph of the feedback nodes and correction terms based on a distance measure on the graph.

The regional FMP algorithm takes into account the *influence region* of each feedback node. Roughly speaking, a feedback node tends to have more influence on the nearby nodes, and thus we choose the correction terms accordingly. In addition, two feedback nodes with overlapping influence regions are considered to have strong interaction, and thus we keep the edge between them in our "thinned" graph of the feedback nodes. Choosing a suitable influence region for each feedback node is an important step, which is still an ongoing research problem. In the section, we describe the regional FMP algorithm assuming we are given a set of feedback nodes and their influence regions.

Figure 4-4(a) shows the underlying graph structure of a Gaussian graphical model. The bigger red circles represent the given feedback nodes while the smaller black circles represent the non-feedback nodes. The influence region of each feedback node is represented by the big pink circle around it. Nodes covered in a pink circle are said to be within the influence region of a feedback node. In the regional FMP algorithm, a feedback node only adds correction terms to nodes in its influence region. Figure 4-4(b) shows the graph formed by the feedback nodes. A solid red line denotes an edge between two feedback nodes having strong interaction. The graph with only these edges is called the *backbone graph*, which is denoted by $\mathcal{B}$. The backbone graph is then our "thinned" graph for solving the inference problem on the feedback nodes. Note that there are also other ways to define the "thinned" backbone graph $J_{\mathcal{B}}$. The best $J_{\mathcal{B}}$ depends on the parameters and structure of $J_{\mathcal{F}}$. Here we just provide one way to make approximations.

The regional FMP algorithm for the model in Figure 4-4(a) has the following steps:

Step 1: Construct 5 extra potential vectors: $\forall p \in \mathcal{F}, \mathbf{h}^p = J_{\mathcal{T},p}$, each corresponding to one feedback node.

Step 2: Perform BP on $\mathcal{T}$ to obtain $P_{ii}^{\mathcal{T}} = (J_{\mathcal{T}}^{-1})_{ii}$ and $\mu_i^{\mathcal{T}} = (J_{\mathcal{T}}^{-1}\mathbf{h}_{\mathcal{T}})_i$ for each node $i \in \mathcal{T}$. With the 5 extra potential vectors, calculate the feedback gains $g_i^1 = (J_{\mathcal{T}}^{-1}\mathbf{h}^1)_i, g_i^2 = (J_{\mathcal{T}}^{-1}\mathbf{h}^2)_i, \ldots, g_i^5 = (J_{\mathcal{T}}^{-1}\mathbf{h}^5)_i$ for each node $i \in \mathcal{T}$ by BP.

Step 3: Obtain a size-5 graph defined on the backbone graph with model parameters $\widehat{J}_{\mathcal{B}}$ and $\widehat{\mathbf{h}}_{\mathcal{B}}$ given by

$$(\widehat{J}_{\mathcal{B}})_{pq} = \begin{cases} J_{pq} - \sum_{j \in \mathcal{N}(p) \cap \mathcal{T}} J_{pj} g_j^q, & \forall p, q \in \mathcal{F},\ (p,q) \in \mathcal{B} \\ 0 & \forall p, q \in \mathcal{F},\ (p,q) \notin \mathcal{B} \end{cases}$$

$$(\widehat{\mathbf{h}}_{\mathcal{B}})_p = h_p - \sum_{j \in \mathcal{N}(p) \cap \mathcal{T}} J_{pj} \mu_j^{\mathcal{T}},\ \forall p \in \mathcal{F}. \tag{4.22}$$

Step 4: Perform BP (LBP) within the backbone graph with information matrix $\widehat{J}_B$ and potential vector $\widehat{\mathbf{h}}_{\mathcal{B}}$. After convergence, we obtain the approximate variances $\bar{P}_{ii}$, $\bar{\mu}_i$, $\forall i \in \mathcal{F}$ and $\bar{P}_{ij}$, $\forall (i,j) \in \mathcal{B}$.

Revise the potential vector on $\mathcal{T}$ by

$$\tilde{h}_i = h_i - \sum_{j \in \mathcal{N}(i) \cap \mathcal{F}} J_{ij} \bar{\mu}_j, \ \forall i \in \mathcal{T}.$$

Step 5: Another round of BP with the revised potential vector $\tilde{\mathbf{h}}_{\mathcal{T}}$ gives the approximate means of nodes on $\mathcal{T}$.

Add correction terms to obtain the approximate variances of nodes in $\mathcal{T}$:

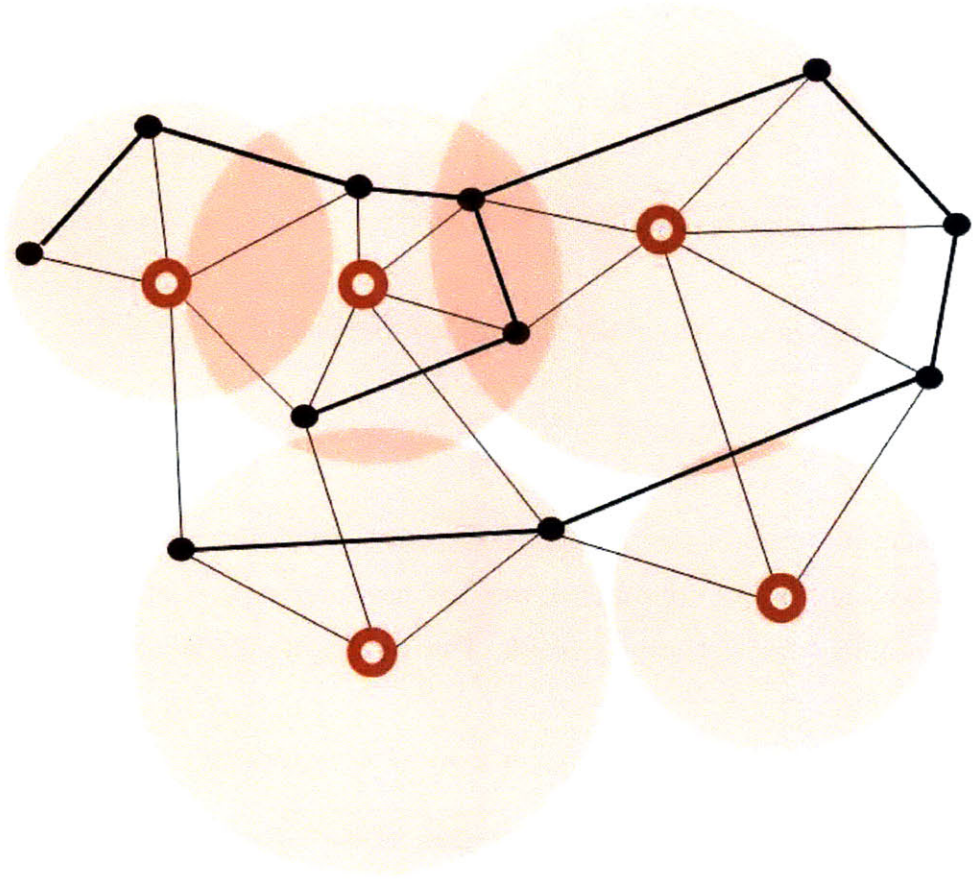
$$P_{ii} = P_{ii}^{\mathcal{T}} + \sum_{p \in \mathcal{F}_i} \sum_{q \in \mathcal{F}_i} g_i^p \bar{P}_{pq} g_i^q, \ \forall i \in \mathcal{T},$$

where $\mathcal{F}_i$ denotes the set of the feedback nodes whose influence region covers node i. If feedback nodes p and q both cover node i, they will have overlapping influence region so that p and q are neighbors in the backbone graph. Thus any $\bar{P}_{pq}$ needed here has been calculated in Step 4.

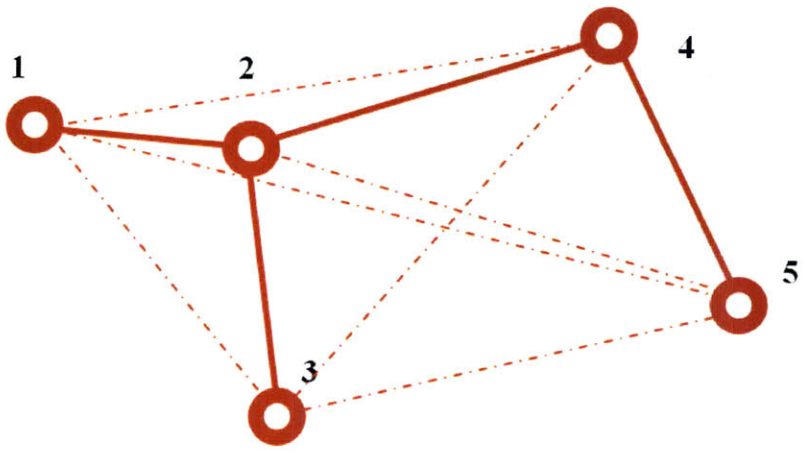
The following proposition gives the complexity of the algorithm.

Proposition 4.8. *In the regional FMP algorithm, if the influence region of each feedback node only overlaps the influence regions of a bounded number of other feedback nodes, the computational complexity of the algorithm is $\mathcal{O}(k(n+D))$, where n is the total number of nodes, k is the size of the FVS, and D is the number of iterations of messages passing within the backbone graph.*

Proof. In Step 1 and Step 2, it talks $\mathcal{O}(k(n-k))$ to perform BP with multiple potential vectors on $\mathcal{T}$. In step 3 the complexity is $\mathcal{O}(kn)$ to obtain the parameters of the backbone graph. In Step 4, it takes $\mathcal{O}(kD)$ computations to perform LBP within the backbone graph. In Step 5, another BP on $\mathcal{T}$ takes $\mathcal{O}(n)$ to compute the means. The complexity of adding correction terms for the variances is $\mathcal{O}(kn)$ since each feedback node only share influence region with a bounded number of other feedback nodes. Therefore, the total complexity is $\mathcal{O}(k(n+D))$. □



(a) A graphical model with the influence regions of the feedback nodes



(b) The backbone graph of the feedback nodes

Chapter 5

Numerical Results

In Chapter 3 we have proved that the FMP algorithm with an FVS always gives the correct variances and means for all nodes. If the size of the FVS is bounded or grows slowly with the size of the graph, this algorithm is efficient. When the size of the FVS is large, we can use a pseudo-FVS instead. In this chapter, we give numerical results for the performance of the approximate FMP algorithm.

Graphical models defined on grid graphs are widely used in computer vision, oceanography/seismic data modeling, and many other applications. Grid graphs are sparse since each node is only connected to at most four neighboring nodes. However, inference problems on grid graphs cannot be solved in linear time exactly due to the existence of many cycles of various lengths. In addition, it is shown that the size of FVS for a grid graph grows linearly with the number of nodes on the grid [27]. Therefore, we need to select a pseudo-FVS of small size.

In our simulations in this chapter, we consider $l \times l$ grid graphs with different values of l. The graph size is then $n = l^2$. Given the size of a grid graph, we randomly generate an information matrix J, which is sparse with respect to the graph. Its nonzero entries are drawn from an *i.i.d.* uniform distribution ranging between -1 and 1. We ensure J is positive definite by adding proper diagonal values. We also generate a potential vector $\mathbf{h}$, whose entries are drawn from an *i.i.d.* uniform distribution ranging between -1 and 1. In

each experiment, we have a positive definite information matrix J and a potential vector $\mathbf{h} \in [-1, 1]^n$. Without loss of generality, we normalize the information matrix to have unit diagonal.

In this chapter, we show numerical results for the convergence and accuracy of the approximate FMP algorithm. We also compare the performance of the algorithm with different pseudo-FVS.

5.1 Convergence of the Approximate FMP Algorithm

In Figure 5-1 and 5-2, we show simulation results for a 10×10 grid graph. Using pseudo-FVS selection algorithm 1 for convergence purpose (see Figure 4-2), we choose one feedback node with the largest score at each step. The remaining graph and its corresponding spectral radius $\rho(\bar{R})$ are shown in the figures, where $\bar{R}$ takes the entrywise absolute values of $R = I - J$ for a normalized J. Note that in the selection algorithm, all tree branches are cleaned up at each iteration so that there are no degree-one nodes in the remaining graph.

LBP does not converge on the original graph. The corresponding spectral radius is $\rho(\bar{R}) = 1.0477$ as shown in Figure 5-1. When one feedback node is chosen, the spectral radius corresponding to the remaining graph becomes 1.0415. After removing one more node from the graph, the spectral radius becomes 0.97249. As we have discussed in Chapter 4, $\rho(\bar{R}) < 1$ is a sufficient condition for LBP to converge on the remaining graph. Thus our approximate FMP algorithm also converges with the two feedback nodes. Since the spectral radius is non-increasing when more nodes are removed, our approximate FMP algorithm still converges if we choose more feedback nodes. In order to give a clear picture of how the algorithm chooses the feedback nodes, we show more results in Figure 5-2.

In all of our experiments on 10×10 grid graphs, we observe that by choosing only a few nodes (at most three empirically) as pseudo-FVS, we can obtain convergence even if LBP on the original graph diverges. Convergence is usually a nontrivial issue for many distributed algorithms. This approach may inspire more effective algorithms to improve convergence.

In our pseudo-FVS selection algorithm 1, we actually use an upper bound on the spectral

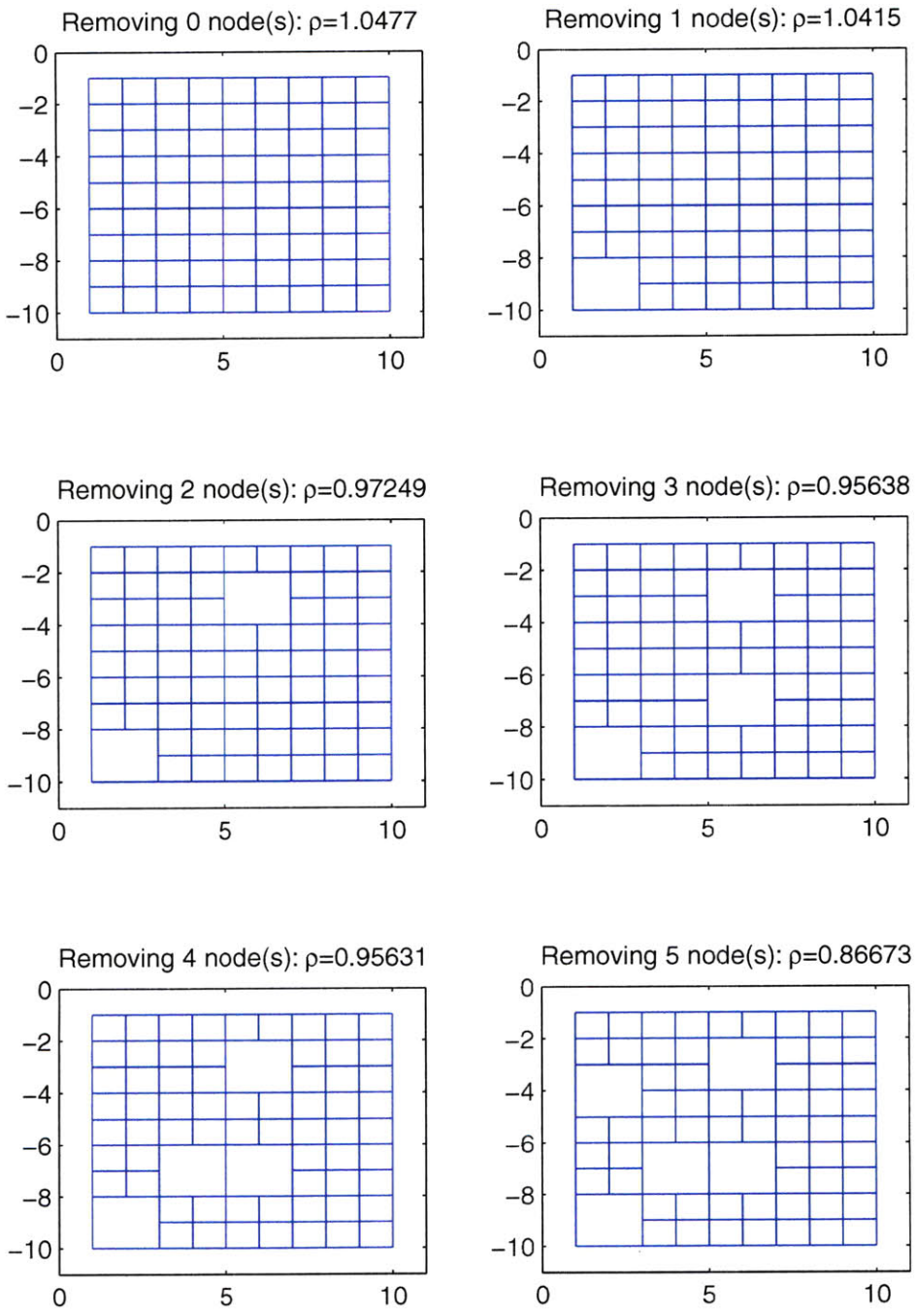


Figure 5-1: Number of removed nodes and the spectral radius—1

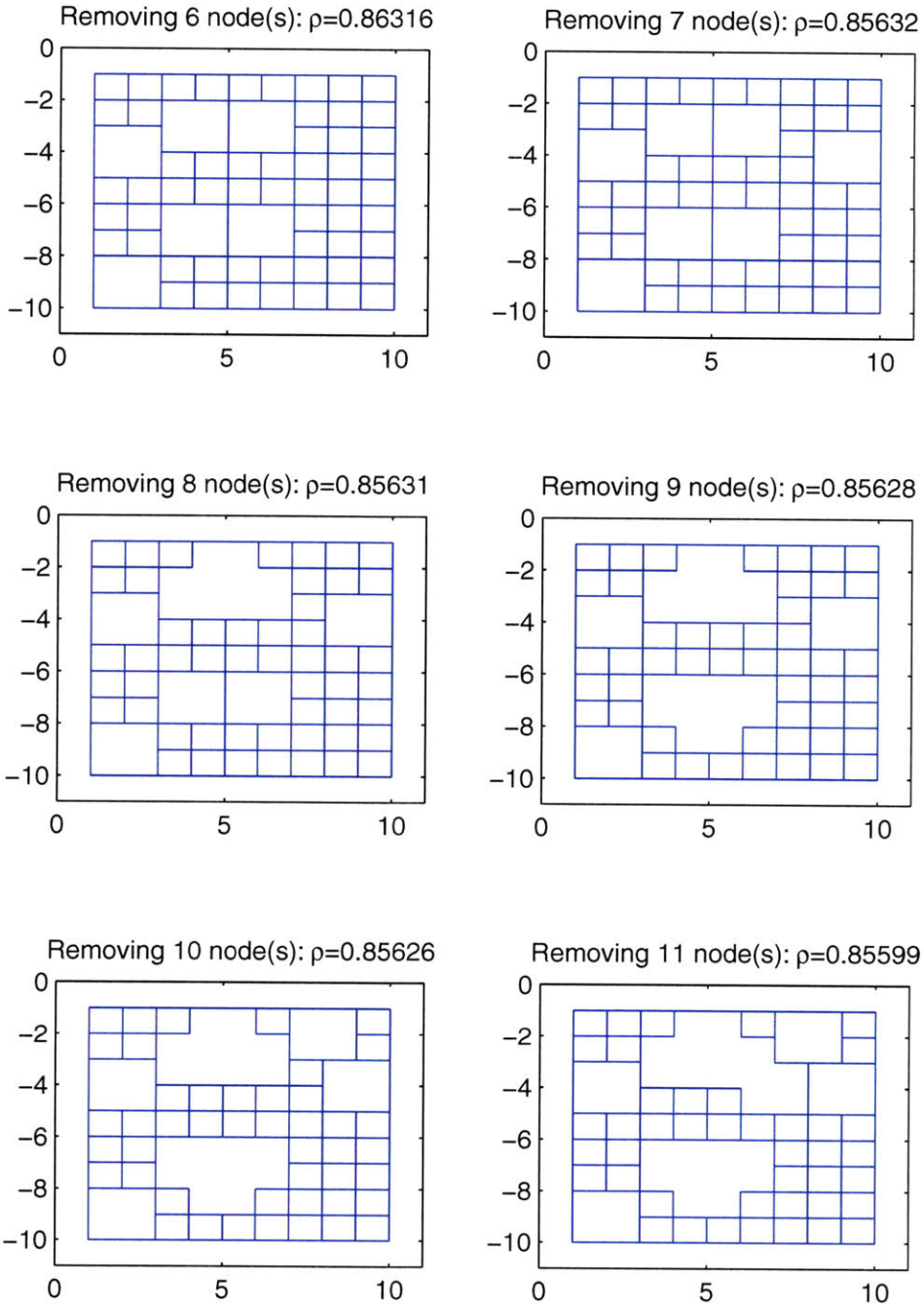
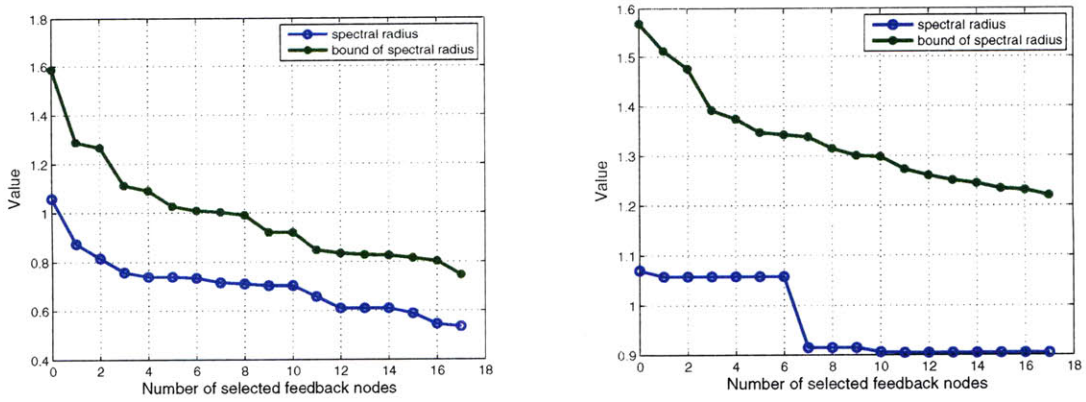


Figure 5-2: Number of removed nodes and the spectral radius—2



(a) 10×10 grid graph

(b) 20×20 grid graph

(c) 40×40 grid graph

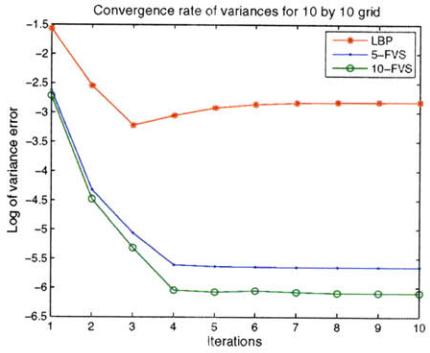
(d) 80×80 grid graph

Figure 5-3: Number of selected feedback nodes v.s. the spectral radius and its bound

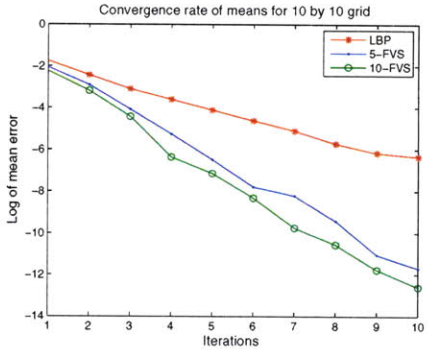
radius instead of computing the exact value. In Figure 5-3 we show how the spectral radius and its upper bound decrease when more and more nodes are selected as feedback nodes. We can see that the spectral radius quickly decreases to be less than one, which is sufficient to give a convergent algorithm.

5.2 Accuracy of the Approximate FMP Algorithm

In this section, we show numerical results for the inference errors. On each grid graph, LBP and the approximate FMP algorithm with two different sets of feedback nodes are performed.

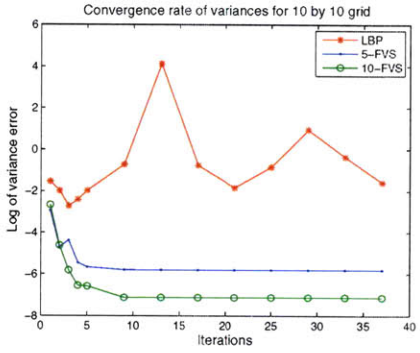


(a) Iterations versus variance errors

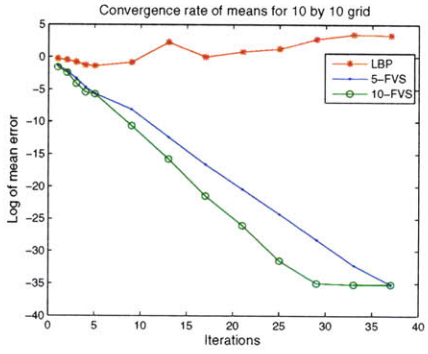


(b) Iterations versus mean errors

Figure 5-4: Inference errors of a 10×10 grid graph



(a) Iterations versus variance errors

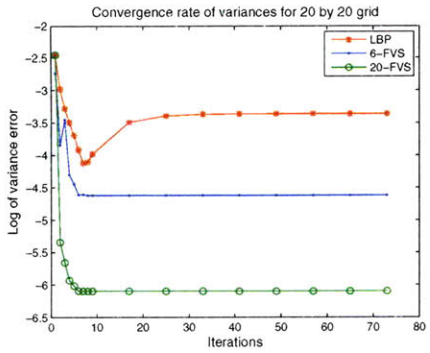


(b) Iterations versus mean errors

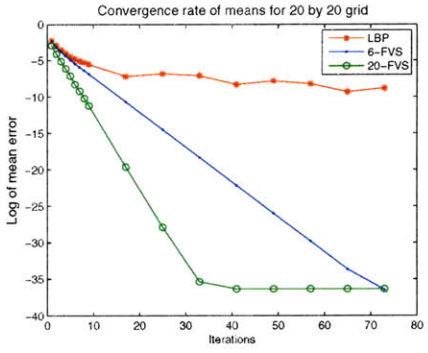
Figure 5-5: Inference errors of a 10×10 grid graph

One set has $k = \lceil \log n \rceil$ feedback nodes while the other has $k = \sqrt{n}$ feedback nodes. The feedback nodes are selected by pseudo-FVS algorithm 2 described in Figure 4-3. The x-axis shows the number of iterations in the message passing steps. The y-axis represents the average absolute errors of all nodes for both variances and means on a logarithmic scale. We use "k-FVS" to denote the approximate FMP algorithm with k feedback nodes.

In Figure 5-4 to 5-8, numerical results are shown for 10×10, 20×20, 40×40 and 80×80 grids respectively. Except for the case shown in Figure 5-4, direct LBP fails to converge on all graphs. With $k = \lceil \log n \rceil$ feedback nodes, the approximate FMP algorithm converges for all grid graphs and gives much better approximations than LBP in fewer iterations. In Figure 5-4 where LBP converges on the original graph, we obtain much less error for the variances

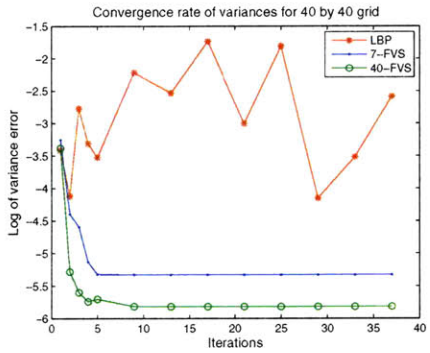


(a) Iterations versus variance errors

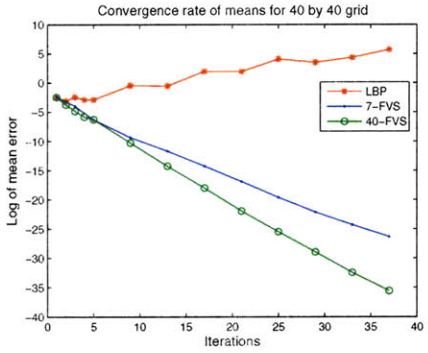


(b) Iterations versus mean errors

Figure 5-6: Inference errors of a 20×20 grid graph

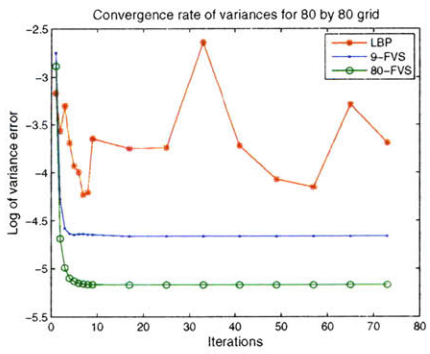


(a) Iterations versus variance errors

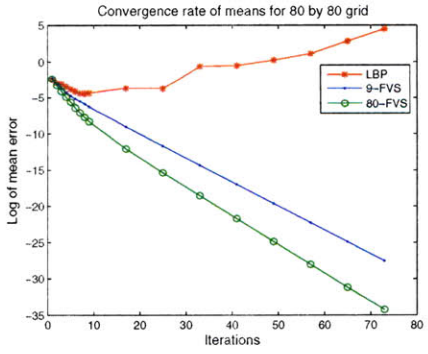


(b) Iterations versus mean errors

Figure 5-7: Inference errors of a 40×40 grid graph



(a) Iterations versus variance errors



(b) Iterations versus mean errors

Figure 5-8: Inference errors of an 80×80 grid graph

and faster convergence rate using our algorithm. In Figure 5-5 to 5-8, direct LBP does not give any meaningful results while our algorithm gives inference results with small errors. When $k = \sqrt{n}$ feedback nodes are used, we obtain even better approximations but with more computations in each iteration. By performing many more experiments, $k = \lceil \log n \rceil$ feedback nodes seem to be sufficient to give a convergent algorithm and good approximations. The complexity of such a method is thus $\mathcal{O}(n \log^2 n)$.

5.3 Choosing a Suitable FVS

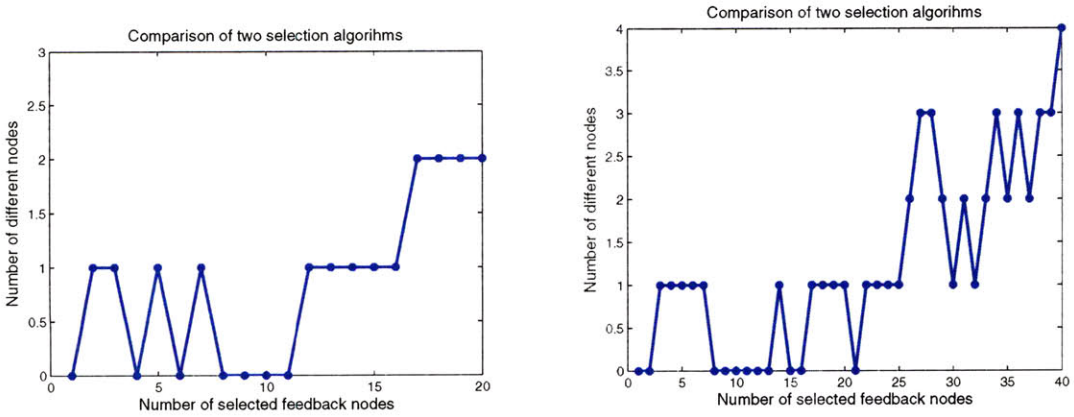
In this section, we compare two pseudo-FVS selection algorithms motivated by convergence and accuracy purposes respectively. These two algorithms are similar in the sense that both of them favor nodes with larger absolute edge weights. In our simulations, the performances are also comparable. This may give us an impression that the size of the pseudo-FVS matters much more than the selection criteria. Actually, numerical results show that the approximate FMP algorithm does not give satisfactory results if we choose a "bad" pseudo-FVS.

We use both algorithms to choose various numbers of feedback nodes for 20×20 and 40×40 graphs respectively. In Figure 5-9, we plot the number of different nodes obtained by the two algorithms. We observe that the algorithms give similar pseudo-FVS, where only about 10% of the selected nodes are different.

In Figure 5-12 we perform the approximate FMP algorithm with a badly selected pseudo-FVS. We can see that LBP, 7-FVS and 40-FVS algorithms all fail to converge. In Figure 5-13 the approximate FMP algorithm is performed with a random pseudo-FVS. In this graphical model, LBP, 7-FVS and 40-FVS algorithms all converge. However, 7-FVS and 40-FVS algorithms give almost the same results as LBP, even though more computations are involved.

These unsatisfactory performances are neither surprising nor discouraging. Actually they suggest that when a suitable set of feedback nodes are selected, we indeed take advantages of the graph structure and model parameters. This is also the high level idea of our algorithm.

As a summary of this chapter, the numerical results seem to suggest that the approximate



(a) 20×20 grid (b) 40×40 grid

Figure 5-9: Comparison of the two FVS selection algorithms

FMP algorithm with only a small number of well selected feedback nodes can give convergent and accurate inference results in a tractable way.

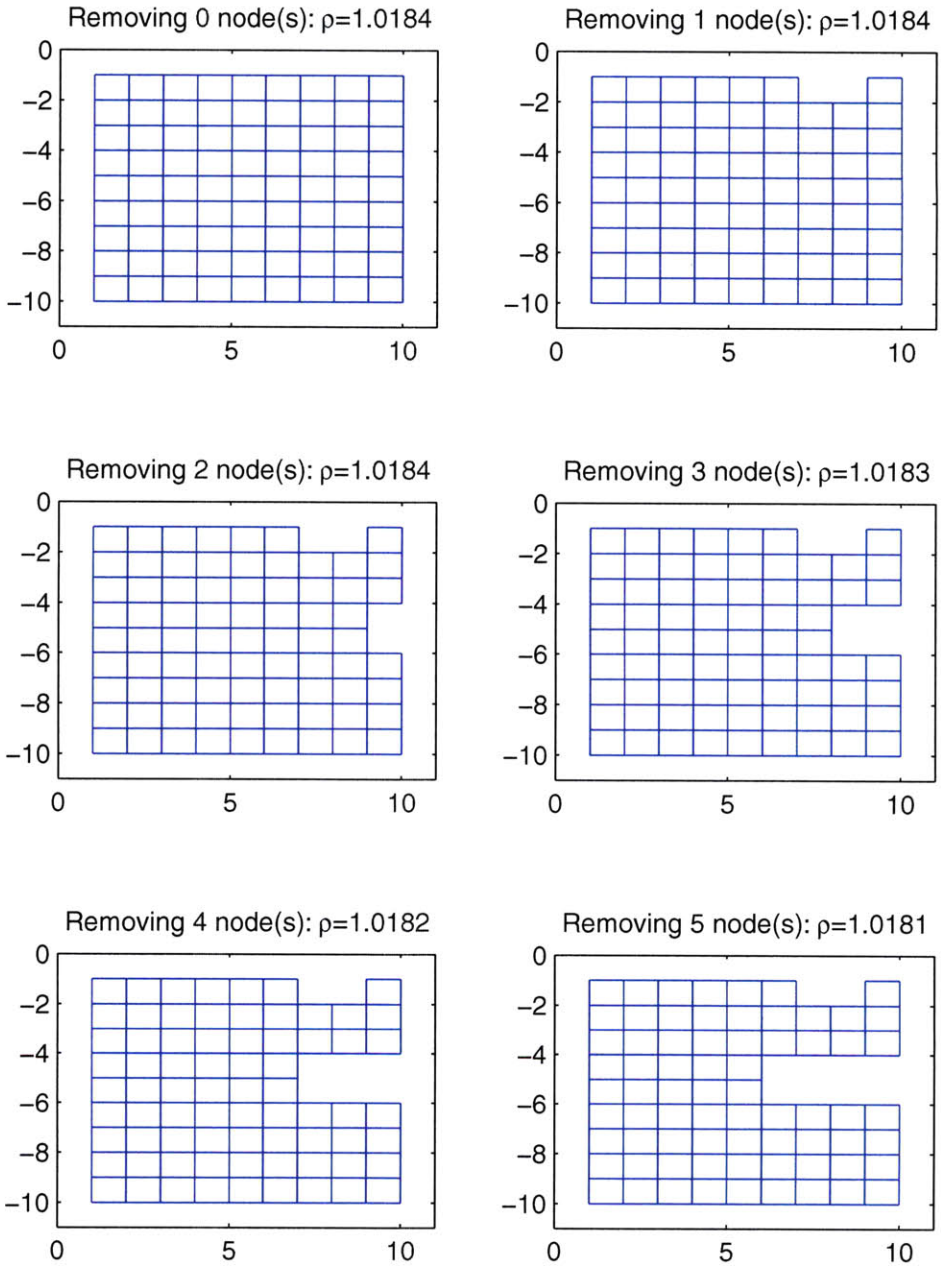
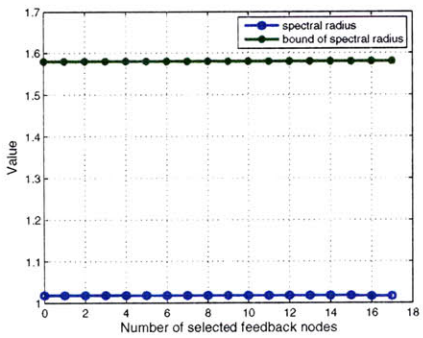
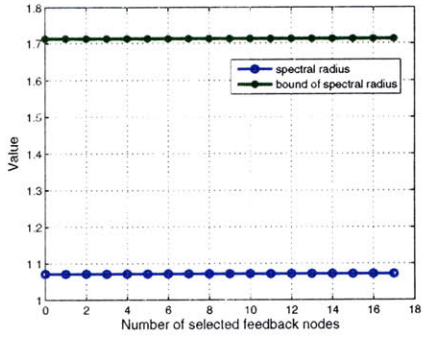


Figure 5-10: The spectral radius and the remaining graph

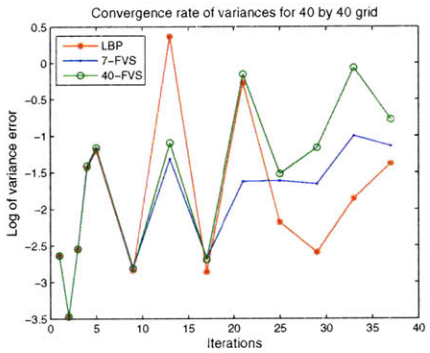


(a) 10×10 grid graph

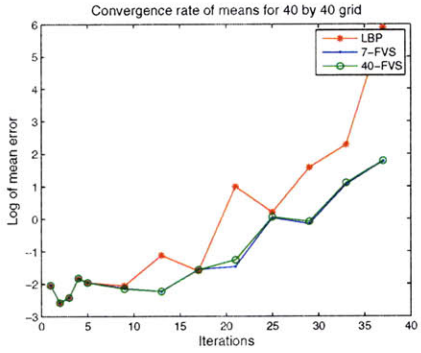


(b) 20×20 grid graph

Figure 5-11: Number of selected feedback nodes v.s. the spectral radius

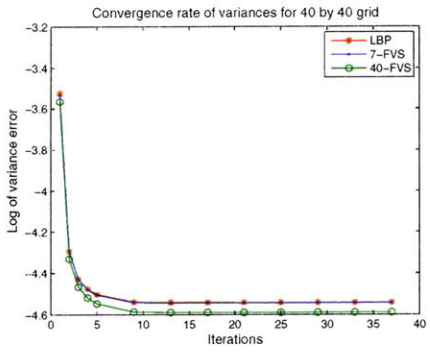


(a) Iterations versus variance errors

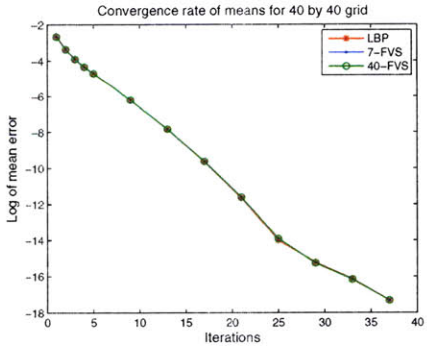


(b) Iterations versus mean errors

Figure 5-12: Inference errors with a bad selection of feedback nodes



(a) Iterations versus variance errors



(b) Iterations versus mean errors

Figure 5-13: Inference errors with a random selection of feedback nodes

Chapter 6

Conclusions

In this thesis we have developed the feedback message passing algorithm where we identify a set of feedback nodes. The high level idea of the algorithm is to use a special message passing scheme for the feedback nodes while using standard BP for other nodes. The feedback message passing algorithm solves inference problems in a Gaussian graphical model in linear time if the graph has a small FVS. For a graph with a large FVS, the approximate feedback message passing algorithm can be used instead. By carefully choosing a small number of feedback nodes, accurate inference results can be obtained efficiently.

In this chapter, we outline the contributions of this thesis and propose future research directions.

6.1 Contributions

Chapter 3.

We introduced the exact FMP algorithm. A special message update scheme is used for the feedback nodes while standard BP is used for the non-feedback nodes. There are two rounds of message passing in the FMP algorithm, where the complexity depends on the number of feedback nodes.

First we describe the algorithm for the single FVS case where a single node can break all

the cycles in the graph. In this case exact inference results can be obtained in linear time. We also give a walk-sum analysis of the correctness of the algorithm.

Then we consider the more general case in which the FVS of a graph has multiple nodes. The complexity for exact inference is $\mathcal{O}(k^2n)$, where k is the size of the FVS and n is the size of the graph.

We show that for a class of graphs with small FVS, inference problems can be solved efficiently by the FMP algorithm. This essentially enlarges the class of graphical models for which exact inference results can be obtained efficiently—from tree-structured graphs to graphs with small FVS.

Chapter 4.

Beyond the class of graphs with small FVS, the approximate FMP algorithm can be applied to graphs with large FVS. We have discussed two ways of making approximations in Chapter 4.

First we discuss the approximate FMP algorithm with a pseudo-FVS, where a pseudo-FVS is a subset of an FVS that does not break all the cycles. Compared to the exact FMP algorithm, the feedback nodes use the same message update scheme, while BP is replaced by LBP for other nodes.

Then we develop some theoretical results. The approximate algorithm is guaranteed to converge if the graphical model is walk-summable. When the algorithm converges, the means obtained by the approximate FMP algorithms capture all necessary walks while the variances capture all backtracking walks plus non-backtrack walks that visit the pseudo-FVS. Different pseudo-FVS selection criteria are proposed for convergence purpose and accuracy purpose based on the theoretical results.

We also briefly introduce the regional FMP algorithm as an alternative way to make approximations.

Chapter 5.

In this chapter, we show experimental results for the convergence and accuracy of the approximate FMP algorithm. We compare the performance of the algorithm with different pseudo-FVS. The numerical results seem to suggest that with only a small number of well selected feedback nodes the algorithm can give accurate inference results.

6.2 Future Research

In this section, we propose several future research directions related to the FMP algorithm.

Improving the Accuracy of Variance Estimate

When the exact FMP algorithm becomes intractable with large FVS, approximations must be made. In this thesis, we have discussed the approximate FMP algorithm with a pseudo-FVS and the regional FMP algorithm. Though these approximations work well in practice, there is still much space for improvement. For instance, there may be better criteria to choose a pseudo-FVS if a tight error bound of variance estimation can be found. In the regional FMP algorithm, how to decide the influence regions and how to obtain the "thinned" backbone graph are still open questions.

In addition, we can also propose other approximation schemes, which could be combinations of previous algorithms and FMP based algorithms.

Computing the Partition Function

Besides the marginal distributions and the most likely state, computing the partition function is another important inference problem for graphical models. For a Gaussian distribution of

$$p(\mathbf{x}) = \frac{1}{Z}\exp\{-\frac{1}{2}\mathbf{x}^T J\mathbf{x} + \mathbf{h}^T\mathbf{x}\},$$

the partition function has a closed form of $Z = \exp\{\frac{1}{2}n\log(2\pi) - \frac{1}{2}\log\det(J) + \frac{1}{2}\mathbf{h}^T J^{-1}\mathbf{h}\}$.

It is not difficult to show that estimating the determinant of the information matrix J accurately is essential to obtain an accurate estimation of the partition function. In [23] the authors use an "orbit-product" framework to analyze the determinant approximation by LBP. It is a natural extension to develop an FMP based algorithm to approximate the determinant. It is interesting to see whether there are any physical interpretations of this kind of approximations as well.

Learning Graphs with Small FVS

In practice our observations of real data are often noisy. The internal structure can be blurred by the noises. A model with simple structure but good generalization power is desirable. Simple structure often suggests efficient inference algorithms. For instance, learning a graphical model of tree structure has been studied [7]. Inference results in tree-structured models can be obtained efficiently by BP.

As we mentioned before, the exact FMP algorithm can solve inference problems of graphs with small FVS efficiently. Thus we have enlarged the class of "simple" models for structure learning. This raises a natural problem of learning a graph with a small FVS that best approximates the empirical distribution. More specifically, we can fix the size of the FVS and learn the best graph structure. Tree-structured graphs can be considered as a special case of graphs with 0-size FVS. It is still an open problem how to find the best structure if the FVS is not known a priori.

Appendix A

An Example of the FMP Algorithm

A.1 BP on a Chain

In this section, we use an example to illustrate the FMP algorithm. We start with a Markov chain, whose inference can be solved by BP exactly. Then we add an extra edge between the first node and the last node on the chain to obtain a single loop of the same size. We apply the FMP algorithm on this loop and compare the messages in the loop with the BP messages in the chain.

The chain graph of n nodes is shown in Figure A-1. The model parameters are J and $\mathbf{h}$. The goal is to compute the mean $\mu_i = (J^{-1}\mathbf{h})_i$ and variance $P_{ii} = (J^{-1})_{ii}$ for each node i. Running BP on the chain is equivalent to performing forward message passing and backward message passing simultaneously. We describe the algorithm as follows. For clarity the message update rules are put into boxes.

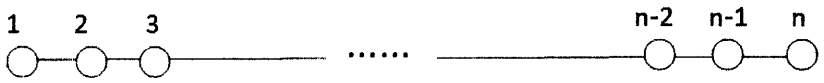


Figure A-1: A chain graphical model

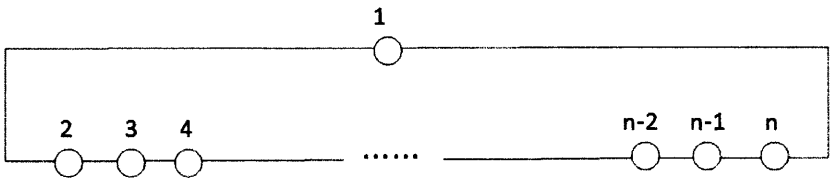


Figure A-2: A single loop graphical model

1. Forward Message Passing

- Initialize

$$\Delta J_{1\to 2} = -J_{12}J_{11}^{-1}J_{12} \quad \text{(A.1)}$$

$$\Delta h_{1\to 2} = -J_{12}J_{11}^{-1}h_1 \quad \text{(A.2)}$$

- For $i = 2, 3, \ldots, n-1$

$$\Delta J_{i\to i+1} = K_i^F J_{i,i+1} \quad \text{(A.3)}$$

$$\Delta h_{i\to i+1} = K_i^F (h_i + \Delta h_{i-1\to i}), \quad \text{(A.4)}$$

where $K_i^F = -J_{i,i+1}(J_{ii} + \Delta J_{i-1\to i})^{-1}$.

2. Backward Message Passing

- Initialize

$$\Delta J_{n\to n-1} = -J_{n,n-1}J_{nn}^{-1}J_{n,n-1} \quad \text{(A.5)}$$

$$\Delta h_{n\to n-1} = -J_{n,n-1}J_{nn}^{-1}h_n \quad \text{(A.6)}$$

- For $i = n-1, n-2 \ldots, 2$

$$\Delta J_{i\to i-1} = K_i^B J_{i,i-1} \quad \text{(A.7)}$$

$$\Delta h_{i\to i-1} = K_i^B (h_i + \Delta h_{i+1\to i}), \quad \text{(A.8)}$$

where $K_i^B = -J_{i,i-1}(J_{ii} + \Delta J_{i+1\to i})^{-1}$.

3. Calculating node marginals

   For $i = 1, 2, \ldots, n$,

$$P_{ii} = (J_{ii} + \Delta J_{i-1\to i} + \Delta J_{i+1\to i})^{-1} \tag{A.9}$$

$$\mu_i = P_{ii}(h_i + \Delta h_{i-1\to i} + \Delta h_{i+1\to i}), \tag{A.10}$$

   where $\Delta J_{0\to 1}, \Delta J_{n+1\to n}, \Delta h_{0\to 1}, \Delta h_{n+1\to n}$ are zeros.

The complexity of the algorithm is $\mathcal{O}(n)$. The exact inference results are computed after the forward message passing step and the backward message passing step. In the two steps, $4(n-4)$ messages are updated.

A.2 The FMP Algorithm on a Single Loop

For a single loop with Gaussian parameters, LBP may not converge. Even when it converges, it gives inaccurate variances. By using the FMP algorithm, *exact* inference on a loop is almost as efficient as on a chain. More specifically, only $8(n-2)$ messages in total are updated for a loop of length n, comparing with $4(n-1)$ messages in BP on a chain of the same size.

A loop of length n is shown in Figure A-2. Any node on the loop forms a feedback vertex set by itself, since the removal of any node breaks the loop and leaves a chain. Without loss of generality, we choose node 1 as the feedback node. The FMP algorithm can be interpreted as performing two rounds of message passing. Both counterclockwise and clockwise messages are passed simultaneously in each round.

1. *First Round of Message Passing*

(a) Counterclockwise Message Passing

- Initialize

$$\Delta J^{\mathcal{T}}_{2\to 3} = -J_{23}J_{22}^{-1}J_{23} \tag{A.11}$$

$$\Delta h^{\mathcal{T}}_{2\to 3} = -J_{23}J_{22}^{-1}h_2 \tag{A.12}$$

$$\Delta h^{1}_{2\to 3} = -J_{23}J_{22}^{-1}J_{12} \tag{A.13}$$

- For $i = 3, 4, \ldots, n-1$

$$\Delta J^{\mathcal{T}}_{i\to i+1} = K_i^{\alpha} J_{i,i+1} \tag{A.14}$$

$$\Delta h^{\mathcal{T}}_{i\to i+1} = K_i^{\alpha}(h_i + \Delta h^{\mathcal{T}}_{i-1\to i}) \tag{A.15}$$

$$\Delta h^{1}_{i\to i+1} = K_i^{\alpha}\Delta h^{1}_{i-1\to i}, \tag{A.16}$$

where $K_i^{\alpha} = -J_{i,i+1}(J_{ii} + \Delta J^{\mathcal{T}}_{i-1\to i})^{-1}$.

(b) Clockwise Message Passing

- Initialize

$$\Delta J^{\mathcal{T}}_{n\to n-1} = -J_{n,n-1}J_{nn}^{-1}J_{n,n-1} \tag{A.17}$$

$$\Delta h^{\mathcal{T}}_{n\to n-1} = -J_{n,n-1}J_{nn}^{-1}h_n \tag{A.18}$$

$$\Delta h^{1}_{n\to n-1} = -J_{n,n-1}J_{nn}^{-1}J_{1n} \tag{A.19}$$

- For $i = n-1, n-2, \ldots, 3$

$$\Delta J^{\mathcal{T}}_{i\to i-1} = K_i^{\beta} J_{i,i-1} \tag{A.20}$$

$$\Delta h^{\mathcal{T}}_{i\to i-1} = K_i^{\beta}(h_i + \Delta h^{\mathcal{T}}_{i+1\to i}) \tag{A.21}$$

$$\Delta h^{1}_{i\to i-1} = K_i^{\beta}\Delta h^{1}_{i+1\to i}, \tag{A.22}$$

where $K_i^{\beta} = -J_{i,i-1}(J_{ii} + \Delta J^{\mathcal{T}}_{i+1\to i})^{-1}$.

(c) Calculating local information

For $i = 2, 3, \ldots, n$,

$$P_{ii}^{\mathcal{T}} = (J_{ii} + \Delta J_{i-1\to i}^{\mathcal{T}} + \Delta J_{i+1\to i}^{\mathcal{T}})^{-1} \tag{A.23}$$

$$g_i^1 = P_{ii}^{\mathcal{T}}(\Delta h_{i-1\to i}^1 + \Delta h_{i+1\to i}^1), \tag{A.24}$$

where $\Delta J_{1\to 2}, \Delta J_{n+1\to n}, \Delta h_{1\to 2}^1, \Delta h_{n+1\to n}^1$ are zeros.

2. *Inference at the Feedback Node* 1

Just for node 1's neighbors node 2 and node n, we calculate the corresponding entries in $\mu^{\mathcal{T}}$ by

$$\mu_2^{\mathcal{T}} = P_{22}^{\mathcal{T}}(h_2 + \Delta h_{3\to 2}^{\mathcal{T}}) \tag{A.25}$$

$$\mu_n^{\mathcal{T}} = P_{nn}^{\mathcal{T}}(h_n + \Delta h_{n-1\to n}^{\mathcal{T}}), \tag{A.26}$$

where $P_{22}^{\mathcal{T}}$, $\Delta h_{3\to 2}^{\mathcal{T}}$ and $\Delta h_{n-1\to n}^{\mathcal{T}}$ have been calculated.

Node 1 then compute its own exact variance and exact mean by

$$P_{11} = (J_{11} - J_{12}g_2^1 - J_{1n}g_n^1)^{-1} \tag{A.27}$$

$$\mu_1 = P_{11}(h_1 - J_{12}\mu_2^{\mathcal{T}} - J_{1n}\mu_n^{\mathcal{T}}), \tag{A.28}$$

where g_2^1 and g_n^1 are computed in Step 1c.

The feedback node 1 revises the node potentials of its neighbors as

$$\widetilde{h}_2 = h_2 - J_{12}\mu_1 \tag{A.29}$$

$$\widetilde{h}_n = h_n - J_{1n}\mu_1. \tag{A.30}$$

Node 1 also creates a message of its own exact variance P_{11} and spreads the message to other nodes. This message can be passed in either clockwise or counterclockwise direction.

3. *Second Round of Message Passing*

(a) Counterclockwise Message Passing

- Initialize

$$\Delta\tilde{h}_{2\to 3} = -J_{23}J_{22}^{-1}\tilde{h}_2 \tag{A.31}$$

- For $i = 3, 4, \ldots, n-1$,

$$\Delta\tilde{h}_{i\to i+1} = K_i^{\alpha}(h_i + \Delta\tilde{h}_{i-1\to i}), \tag{A.32}$$

where K_i^{α} is the same K_i^{α} as in the first round of message passing.

(b) Counterclockwise Message Passing

- Initialize

$$\Delta\tilde{h}_{n\to n-1} = -J_{n,n-1}J_{nn}^{-1}\tilde{h}_n \tag{A.33}$$

- For $i = n-1, n-2, \ldots, 3$,

$$\Delta\tilde{h}_{i\to i-1} = K_i^{\beta}(h_i + \Delta\tilde{h}_{i+1\to i}), \tag{A.34}$$

where K_i^{β} is the same K_i^{β} as in the first round of message passing.

(c) Calculating the exact means and variances of node in $\mathcal{T}$

For $i = 2, 3, \ldots, n$,

$$\mu_i = P_{ii}^{\mathcal{T}}(\Delta\tilde{h}_{i-1\to i} + \Delta\tilde{h}_{i+1\to i}), \tag{A.35}$$

where $\Delta\tilde{h}_{1\to 2}, \Delta\tilde{h}_{n+1\to n}$ are zeros and $P_{ii}^{\mathcal{T}}$ are calculated in Step 1c.

The exact variances can be computed by adding correction terms to $P_{ii}^{\mathcal{T}}$.

For $i = 2, 3, \ldots, n$,

$$P_{ii} = P_{ii}^{\mathcal{T}} + P_{11}(g_i^1)^2, \tag{A.36}$$

where P_{11} has been passed to every node in the loop and g_i^1 are computed in Step

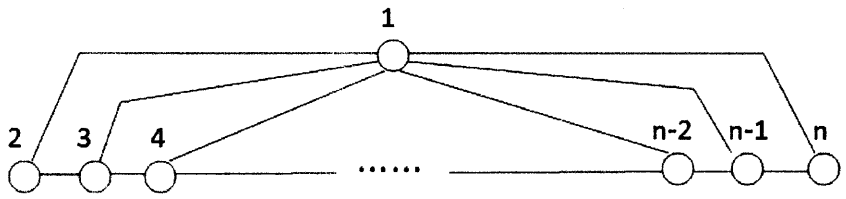


Figure A-3: A graphical model with many loops

1c.

In the whole process of the FMP algorithm, only $8(n-2)$ messages are updated and passed locally, which involve $\Delta J_{i\to j}$, $\Delta h^{\mathcal{T}}_{i\to j}$, $\Delta h^{1}_{i\to j}$ and $\Delta \tilde{h}_{i\to j}$ for neighboring i and j. Therefore, the inference problem can be solved efficiently for a single loop. Beyond a single loop, inference on many other graphs with a single feedback node can be solved by the FMP algorithm in the same way. For example, node 1 in Figure A-3 is connected to all the other nodes. Any two nodes in the graph can reach each other within two steps. Although there are many cycles in the graph, exact inference is still efficient since we can choose node 1 as the feedback node and run the FMP algorithm similarly.

Bibliography

[1] V. Bafna, P. Berman, and T. Fujito, "A 2-approximation algorithm for the undirected feedback vertex set problem," *SIAM Journal on Discrete Mathematics*, vol. 12, p. 289, 1999.

[2] R. Bar-Yehuda, D. Geiger, J. Naor, and R. Roth, "Approximation algorithms for the vertex feedback set problem with applications to constraint satisfaction and Bayesian inference," in *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1994, pp. 344–354.

[3] M. Beal, F. Falciani, Z. Ghahramani, C. Rangel, and D. Wild, "A Bayesian approach to reconstructing genetic regulatory networks with hidden factors," *Bioinformatics*, vol. 21, no. 3, p. 349, 2005.

[4] G. Blin, F. Sikora, and S. Vialette, "Querying protein-protein interaction networks," in *Proceedings of the 5th International Symposium on Bioinformatics Research and Applications*. Springer, 2009, pp. 52–62.

[5] S. Cagnoni, A. Dobrzeniecki, R. Poli, and J. Yanch, "Genetic algorithm-based interactive segmentation of 3D medical images," *Image and Vision Computing*, vol. 17, no. 12, pp. 881–895, 1999.

[6] V. Chandrasekaran, J. Johnson, and A. Willsky, "Estimation in Gaussian graphical models using tractable subgraphs: a walk-sum analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 5, p. 1916, 2008.

[7] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.

[8] C. Crick and A. Pfeffer, "Loopy belief propagation as a basis for communication in sensor networks," in *Uncertainty in Artificial Intelligence*, vol. 18, 2003.

[9] F. Dehne, M. Fellows, M. Langston, F. Rosamond, and K. Stevens, "An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem," *Theory of Computing Systems*, vol. 41, no. 3, pp. 479–492, 2007.

[10] R. Diestel, "Graph theory."

[11] H. El Gamal and A. Hammons, "Analyzing the turbo decoder using the Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 671–686, 2001.

[12] P. Erdős and L. Pósa, "On the maximal number of disjoint circuits of a graph," *Publicationes Mathematicae Debrecen*, vol. 9, pp. 3–12, 1962.

[13] P. Fieguth, W. Karl, A. Willsky, and C. Wunsch, "Multiresolution optimal interpolation and statistical analysis of TOPEX/POSEIDON satellite altimetry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 2, pp. 280–292, 1995.

[14] F. Fomin, S. Gaspers, A. Pyatkin, and I. Razgon, "On the minimum feedback vertex set problem: exact and enumeration algorithms," *Algorithmica*, vol. 52, no. 2, pp. 293–307, 2008.

[15] J. Fry and E. Gaztanaga, "Biasing and hierarchical statistics in large-scale structure," *The Astrophysical Journal*, vol. 413, p. 447, 1993.

[16] D. Geiger and C. Meek, "Graphical models and exponential families," in *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*. Citeseer, 1998, pp. 156–165.

[17] D. Heckerman and J. Breese, "Causal independence for probability assessment and inference using Bayesian networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 26, no. 6, pp. 826–831, 1996.

[18] P. Holland and S. Leinhardt, "An exponential family of probability distributions for directed graphs," *Journal of the American Statistical Association*, vol. 76, no. 373, pp. 33–50, 1981.

[19] R. Horn and C. Johnson, *Matrix analysis*. Cambridge University Press, 1990.

[20] A. Ihler, J. Fisher, and A. Willsky, "Loopy belief propagation: Convergence and effects of message errors," *Journal of Machine Learning Research*, vol. 6, no. 1, p. 905, 2006.

[21] M. Jeruchim, P. Balaban, and K. Shanmugan, *Simulation of communication systems: modeling, methodology, and techniques*. Springer, 2000.

[22] J. Johnson, D. Bickson, and D. Dolev, "Fixing convergence of Gaussian belief propagation," in *International Symposium on Information Theory*.

[23] J. Johnson, V. Chernyak, and M. Chertkov, "Orbit-product representation and correction of Gaussian belief propagation," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 473–480.

[24] M. Jordan, "Graphical models," *Statistical Science*, pp. 140–155, 2004.

[25] R. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, vol. 43, pp. 85–103, 1972.

[26] D. Kratsch, H. Müller, and I. Todinca, "Feedback vertex set on AT-free graphs," *Discrete Applied Mathematics*, vol. 156, no. 10, pp. 1936–1947, 2008.

[27] F. Madelaine and I. Stewart, "Improved upper and lower bounds on the feedback vertex numbers of grids and butterflies," *Discrete Mathematics*, vol. 308, no. 18, pp. 4144–4164, 2008.

[28] D. Malioutov, "Approximate inference in Gaussian graphical models," Ph.D. dissertation, Massachusetts Institute of Technology, 2008.

[29] D. Malioutov, J. Johnson, and A. Willsky, "Walk-sums and belief propagation in Gaussian graphical models," *Journal of Machine Learning Research*, vol. 7, pp. 2031–2064, 2006.

[30] K. Mardia, "Multi-dimensional multivariate Gaussian Markov random fields with application to image processing," *Journal of Multivariate Analysis*, vol. 24, no. 2, pp. 265–284, 1988.

[31] R. McEliece, D. MacKay, and J. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, 1998.

[32] K. Murphy, Y. Weiss, and M. Jordan, "Loopy belief propagation for approximate inference: an empirical study," in *Proceedings of Uncertainty in Artificial Intelligence*, 1999, pp. 467–475.

[33] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[34] S. Roy, "Synthesizing designs with low-cardinality minimum feedback vertex set for Partial Scan Application," in *IEEE VLSI Test Symposium*. IEEE Computer Society, 1994, p. 2.

[35] A. Shamir, "A linear time algorithm for finding minimum cutsets in reducible graphs," *SIAM Journal on Computing*, vol. 8, p. 645, 1979.

[36] M. Sonka, V. Hlavac, and R. Boyle, "Image processing, analysis, and machine vision second edition," 1999.

[37] E. Sudderth, M. Wainwright, and A. Willsky, "Embedded trees: estimation of Gaussian processes on graphs with cycles," *IEEE Transactions on Signal Processing*, vol. 52, no. 11, pp. 3136–3150, 2004.

[38] S. Tatikonda and M. Jordan, "Loopy belief propagation and Gibbs measures," in *Uncertainty in Artificial Intelligence*, vol. 18, 2002, pp. 493–500.

[39] V. Vazirani, *Approximation algorithms*. Springer, 2004.

[40] M. Wainwright and M. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.

[41] C. Wang, E. Lloyd, and M. Soffa, "Feedback vertex sets and cyclically reducible graphs," *Journal of the ACM*, vol. 32, no. 2, pp. 296–313, 1985.

[42] Y. Weiss and W. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, no. 10, pp. 2173–2200, 2001.

[43] A. Werhli, M. Grzegorczyk, and D. Husmeier, "Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical Gaussian models and Bayesian networks," *Bioinformatics*, vol. 22, no. 20, p. 2523, 2006.

[44] A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. Von Rohr, L. Thiele *et al.*, "Sparse graphical Gaussian modeling of the isoprenoid gene network in Arabidopsis thaliana," *Genome Biology*, vol. 5, no. 11, p. R92, 2004.

[45] C. Wunsch and P. Heimbach, "Practical global oceanic state estimation," *Physica D: Nonlinear Phenomena*, vol. 230, no. 1-2, pp. 197–208, 2007.

[46] L. Yang, X. Liu, C. Jursa, M. Holliman, A. Rader, H. Karimi, and I. Bahar, "iGNM: a database of protein functional motions based on Gaussian Network Model," *Bioinformatics*, vol. 21, no. 13, p. 2978, 2005.