EFFICIENT INFERENCE FOR HYBRID BAYESIAN NETWORKS

by

Wei Sun
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

[illegible]	Dr. KC Chang, Dissertation Director
[illegible]	Dr. Kathryn Blackmond Laskey, Committee Member
[illegible]	Dr. Kristine Bell, Committee Member
[illegible]	Dr. James Gentle, Committee Member
[illegible]	Dr. Daniel Menascé, Associate Dean for Research and Graduate Studies
[illegible]	Dr. Lloyd J. Griffiths, Dean, The Volgenau School of Information Technology and Engineering

Date: Nov. 27, 2007

Fall Semester 2007
George Mason University
Fairfax, VA

Efficient Inference For Hybrid Bayesian Networks

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Wei Sun

Master of Science in Operations Research
George Mason University, May 2003

Bachelor of Science in Electrical Engineering
Zhejiang University, China, July 1991

Director: Dr. KC Chang, Professor
Department of Systems Engineering and Operations Research

Fall Semester 2007
George Mason University
Fairfax, VA

Acknowledgments

First and foremost, my sincere gratitude goes to my advisor Dr.KC Chang. It is my great luck to know him and have him as my supervisor, mentor, and eventually reliable friend. His work ethic, academic integrity and inspirational vision have encouraged and guided me through my study and research journey at GMU these years. He is always available when I need guidance. In the mean time, he gave me the maximum freedom to explore and develop. One thing for sure to me is that without his help, this dissertation is not possible.

I would like to thank my committee members: Dr.Kathryn Laskey, Dr.Kristine Bell and Dr.James Gentle. It is my honor and pleasure to have the chance to learn from them. In particular, Dr.Laskey has taught me a lot about rigorous thinking, intuitive modeling in her class and the student seminar organized by her as well. Dr.Bell led me into the wonderful world of statistics and Dr.Gentle has shown me his excellence in computational statistics. I really appreciate your insightful suggestions, generous help and warm encouragement throughout the completion of this dissertation.

Special thanks are due to Dr.Karla Hoffman. It is she who provided me an admission and assistantship when I applied for the graduate program in operations research at GMU. And it is she who led me the first step into the theoretical OR field when I took her class OR541 in the first semester of my GMU study. Afterward, she is always helpful and responsive for my academic queries and general questions as well. She may not know how grateful I am for her instructions.

Finally and above all, I like to take this opportunity to express my deep appreciation to my beloved parents and my wife Li, for their endless love and consistent support. Also I would like to thank my brother and sister for their efforts to take care of the family while I pursued my graduate study overseas. I love you! This dissertation is dedicated to them.

Table of Contents

List of Tables

List of Figures

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
BN	Bayesian Network
CHM	Conditional hybrid Bayesian network model
CLG	Conditional Linear Gaussian
CPD	Conditional Probability Distribution
CPT	Conditional Probability Table
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network
GMU	George Mason University
HMM	Hidden Markov Model
ICPD	Important Conditional Probability Distribution
ICPT	Important Conditional Probability Table
ID	Influence Diagram
PDBN	Partial Dynamic Bayesian Network
PF	Particle Filter
UKF	Unscented Kalman Filter

Abstract

EFFICIENT INFERENCE FOR HYBRID BAYESIAN NETWORKS
Wei Sun, PhD
George Mason University, 2007
Dissertation Director: Dr. KC Chang

Uncertainty is everywhere in real life so we have to use stochastic model for most real-world problems. In general, both the systems mechanism and the observable measurements involve random noise. Therefore, probability theory and statistical estimation play important roles in decision making. First of all, we need a good knowledge representation to integrate information under uncertainty; then we need to conduct efficient reasoning about the state of the world given noisy observations.

Bayesian networks (BNs) provide a compact, efficient and easy-to-interpret way to model the joint probability distribution of random variables over a problem domain. A Bayesian network encodes dependency relationship between random variables into a graphical probabilistic model. The structural properties and expressive power of Bayesian network make it an excellent knowledge base for effective probabilistic inference. Over the past several decades, a number of exact and approximate inference algorithms have been proposed and applied for inference in different types of Bayesian networks. However, in general, BN probabilistic inference is NP-hard. In particular, probabilistic reasoning for BNs with nonlinear non-Gaussian hybrid model is known to be one of the most difficult problems. First, no exact method is possible to compute the posterior distributions in such case. Second, relatively little research has been done for general hybrid models. Unfortunately, most real-world problems are naturally modeled with both categorical variables and continuous variables with typically nonlinear relationship.

This dissertation focuses on the hybrid Bayesian networks containing both discrete and continuous random variables. The hybrid model may involve nonlinear functions in conditional probability distributions and the distributions could be arbitrary. I first give a thorough introduction to Bayesian networks and review of the state-of-the-art inference algorithms in the literature. Then a suite of efficient algorithms is proposed to compute the posterior distributions of hidden variables for arbitrary continuous and hybrid Bayesian networks. Moreover, in order to evaluate the performance of the algorithms with hybrid Bayesian networks, I present an approximate analytical method to estimate the performance bound. This method can help the decision maker to understand the prediction performance of a BN model without extensive simulation. It can also help the modeler to build and validate a model effectively. Solid theoretical derivations and promising numerical experimental results show that the research in this dissertation is fundamentally sound and can be applied in various decision support systems.

Chapter 1: Introduction

1.1 Motivation

Consider the following problem:

> A mortgage company likes to know the default probability for an individual who applies for a loan. They have vast amounts of historical data including customer's information such as age, financial behavior, payment history, credit score and macro-economical data such as interest rate, employment index at the corresponding period, etc. The question is which prediction model should they use to fulfill this task?

Conventional statisticians usually build a logit or probit model for predicting the probability if the dependent variable, also called exposure or response variable, is binomial or multinomial. These models require a fixed form of link function. Then from data, they usually estimate the corresponding parameters using a maximum likelihood estimation method. When using the model to compute the predicted probability, they need to know the values of all independent variables. In general, all parametric regression models require that the dependent variable is a function of all independent variables and typically can only handle a limited number of variables. However, it is usually difficult to estimate a function, especially when there might be many variables involved. And most of the time the random variables interact with each other; for example, some variables may have causal relationship. Moreover, not all independent variables are observable.

Along with the fast growing computing power and rapid accumulation of data in recent decades, machine learning techniques have been applied in discovering hidden pattern in voluminous databases. Artificial neural network (ANN) [JMM96, Hay98], a nonparametric model originally inspired by biological networks of neurons in the human brain, is an adaptive procedure to model the relationship between input and output variables without any specific structure. It is free from the assumption of normality or linearity. Although ANN can explore complex nonlinear structures, its "Black-box" mechanism learned at an "unconscious" level and its free parameters without physical meaning related to problem make it difficult to understand and interpret the results.

Fortunately, another powerful probabilistic modeling tool called Bayesian network (BN) [Pearl88, Jensen96] emerged in early 1980s in the field of artificial intelligence (formal definition and thorough introduction of BNs will be given in Chapter 2). It is being called "Bayesian" originally because Bayes' rule was used to compute the posterior probability distribution of variables of interest given observations. A Bayesian network achieves the task of modeling complex relationship between variables by decomposing the joint probability distribution of random variables into local conditional distributions. It is not necessary in BNs to model one variable depending on all other variables like statistical regression-based model does. Instead, only variables having direct dependence relationships are modeled as parent and child nodes in a graphical framework, and the local dependence relationship between child and its parents could be learned by using regression-based methods individually. The intrinsic interactions between variables (dependency and independency) are explicitly expressed by the network structure and local conditional probability distributions.

A Bayesian network is a directed acyclic graph (DAG) in which nodes represent random variables, directed arcs between nodes represent dependence relationships, mathematically described by conditional probability distributions (CPDs). The network structure and CPDs could be learned from data as well as elicitated from the domain expert. BNs have the transparent architecture for knowledge representation and probabilistic reasoning. So the inference results provided by BNs are interpretable to obtain insights. As a modeling tool, BNs are not limited to predefined structure and number, type, or relationship of variables. In last several decades, development of machine learning methods and inference algorithms has made BN popular in solving realistic problems. Subsequently, BN has proven its power in many applications, such as data fusion, space navigation, medical diagnosis, fraud detection, software engineering, visual tracking, etc. For an overview and comprehensive description of real-world BN applications, see [Haddawy99] and the March 1995 special issue of Communication of ACM [CACM95].

Bayesian inference as an important statistical method has been used in many fields since Bayesian probability theory was proposed about 250 years ago. To illustrate, let us take a look at a very simple example:

> From historical data, we know the weight of 'Fuji' apple is normally distributed with mean 350 grams and variance 10 $(grams)^2$. John picked up a 'Fuji' from the tree in the farm. He wants to know the weight of this apple. But he only has a bad scale available, which is very inaccurate with measurement variance to 20 $(grams)^2$. The bad scale showed 330

grams when weighing the apple. What do you think the true weight of this apple is?

The mathematical model for this problem is: X, represents the weight of 'Fuji' apple, has a prior distribution $P(X)$ as Normal$(350, 10)$; Y, represents the measured weight, has a conditional probability distribution $P(Y|X)$ as Normal$(X, 20)$. A simple two-node Bayesian networks in Figure 1.1 illustrates this relationship. The query we need to answer in the problem is equivalent to computing the posterior distribution $P(X|Y = 330)$. Applying Bayes' rule, it is straightforward to show that:

$$\begin{aligned} P(X|Y=330) &= \frac{P(Y=330|X)P(X)}{P(Y=330)} \\ &= \frac{\frac{1}{\sqrt{2\pi\times 20}}\, e^{-\frac{(330-X)^2}{2\times 20}}\ \frac{1}{\sqrt{2\pi\times 10}}\, e^{-\frac{(X-350)^2}{2\times 10}}}{\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\times 20}}\, e^{-\frac{(330-X)^2}{2\times 20}}\ \frac{1}{\sqrt{2\pi\times 10}}\, e^{-\frac{(X-350)^2}{2\times 10}}\, dx} \\ &= \frac{1}{\sqrt{2\pi V}}\, e^{-\frac{(X-u)^2}{2\times V}} \end{aligned} \tag{1.1}$$

where $u \approx 343.33$, and $V \approx 6.667$. Therefore, given the noisy measurement 330, the updated belief about the apple weight is a Normal random number with mean 343.33 and variance 6.667.

We can find the exact analytical answer in Equation 1.1 because it is a linear Gaussian problem. When nonlinear function is involved between variables, and/or random variables have distributions other than Gaussian, it is very difficult or impossible to find closed-form analytical solution.

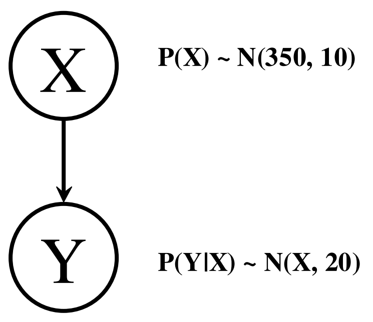


Figure 1.1: ‘Fuji’ Apple Weighing Problem

Moreover, most real-world decision problems modeled by BNs naturally have both discrete and continuous variables. In modern applications, the networks become bigger and more complex. Some can easily have hundreds or even thousands of variables with typically nonlinear relationships [LM-etc02]. The nonlinearity and hybrid natures of BN models make the inference computation very difficult. In general, both exact and approximate inference for Bayesian networks are NP-hard. However, for specific classes of BNs such as pure discrete networks or linear Gaussian continuous networks, many efficient inference algorithms have been developed. For general hybrid models, only approximate methods are possible. So far, relatively little research has been done in this area. Practitioners usually discretize continuous variables, linearize the nonlinear functions, simplify the model by removing weak arcs, or conduct Morte Carlo simulations for hybrid model inference. Obviously, these approaches either lose fidelity due to model simplification or incur significant cost due to extensive simulation.

1.2 Research Objectives and Contribution

This dissertation focuses on the inference problems for hybrid Bayesian networks. Our goal is to develop efficient approximate inference algorithms that perform acceptably on problems with nonlinearity and heterogeneity. So far in the literature, researchers have proposed some algorithms for hybrid BN inference. They include methods using numerical integration [YD06], density approximation techniques such as Gaussian mixture [Poland94, Shenoy06], mixture of truncated exponential [CS06], and importance sampling [YD07]. We know numerical integration are computational intensive with high dimensional problems, and function estimations involve learning process. In this dissertation, we adopted different approaches based on the framework of message passing originally proposed by Pearl [Pearl88]. In our algorithms, estimation techniques such as unscented transformation, function estimation, Gaussian mixture model are integrated in a unified manner.

In summary, the major contribution of this dissertation is twofold. The first is for the algorithmic benefit: we have developed two novel efficient algorithms for hybrid BNs. The other is for theoretical benefit: we have derived an approximate analytical method to estimate the performance bound for model evaluation. Specific research achievements are the following:

- When continuous variables are present in the BNs, because their dependence relationships could be nonlinear and their probability distributions could be arbitrary, no exact inference is possible. One of the most popular methods in this case is stochastic sampling such as Likelihood Weighting algorithm. But with unlikely evidence, simulation methods could be very slow to converge. We propose an efficient approximate inference algorithm called Unscented Message Passing (UMP-BN) [SC07a] for Bayesian network with arbitrary continuous variables. UMP-BN combines unscented transformation—a deterministic sampling method, and Pearl's message passing algorithm to provide estimates of the first two moments of the posterior distributions. We test this algorithm with several networks including one with nonlinear relationships and/or non-Gaussian variables. The numerical experiments show that UMP-BN converges very fast and produces promising results.

- When mixed random variables (continuous and discrete variables) are present in the Bayesian network, there is no theoretical sound method so far for efficient message passing. Based on UMP-BN, we propose a novel approach, called Hybrid Message Passing (HMP-BN), to compute, propagate and integrate the messages for hybrid models [SC07b]. Specifically, we first partition the network into separate parts by introducing the concept of interface nodes. Then different inference algorithm can be applied for each sub-network. Finally HMP-BN integrates the information through the channel of interface nodes and then calculate the posterior distributions for all hidden variables. This algorithm uses the concept of conditioning for problem decomposition. For decision problems, which usually have few discrete variables with many continuous factors, our numerical experiments show that the algorithm is very efficient and accurate in choosing the right decision.

- The accuracy of the computed posterior probability distributions provided by an inference algorithm is essential since the correct decision under a partially observable environment depends on this distribution. However, there is no general evaluation methodology available to predict the inference performance for a BN other than extensive Monte Carlo simulation methods. This dissertation also presents an approximate analytical method to estimate the performance bound [CS04]. It can help the decision maker to understand the model prediction performance without extensive simulation and help the modeler to build and validate the model effectively.

- Most work in this dissertation research has been implemented in MATLAB using Bayesian Network Toolbox (BNT) [Mur01].[1] Therefore, as a by-product, the algorithms developed in the dissertation have been added in the BNT as alternative inference engines. It is accessible for further research from my personal webpage: `http://mason.gmu.edu/~wsun/research.htm`.

1.3 Dissertation Overview

The remainder of this dissertation is organized as the follows. Chapter 2 gives a thorough introduction to Bayesian networks. We classify BNs into different types and review the state-of-the-art inference algorithms available in the literature. Chapter 3 describes the message passing algorithm for arbitrary continuous BNs in detail. For general hybrid BNs, Chapter 4 first introduces the network partition schemes. And then message integration procedure between discrete and continuous variables is presented. We summarize the hybrid message passing algorithm with numerical experiments and discuss the complexity of the algorithm at the end of Chapter 4. Chapter 5 derives an approximate analytical method to provide performance bound.

[1]Thanks to Kevin Murphy, etc. for developing and maintaining BNT.

Finally, in Chapter 6, we conclude the dissertation and discuss several potential future research directions.

Conference and journal papers published/submitted about the research in this dissertation are listed as below:

- Wei Sun and KC Chang. *Message Passing for General Bayesian Networks: Representation, Propagation and Integration.* Submitted to IEEE Transactions on Aerospace Electronic Systems. September, 2007.

- Wei Sun and KC Chang. *Convergence Study of Message Passing in Arbitrary Continuous Bayesian Networks.* To appear in SPIE Conference, Orlando, March, 2008.

- Wei Sun and KC Chang. *Hybrid Message Passing for Mixed Bayesian Networks.* In Proceedings of the 10th International Conference on Information Fusion, Quebec, Canada, July 2007.

- Wei Sun and KC Chang. *Unscented Message Passing for Arbitrary Continuous Variables in Bayesian Networks.* In Proceedings of the 22nd AAAI Conference on Artificial Intelligence, Vancouver, Canada, July 2007.

- Wei Sun and KC Chang. *Probabilistic Inference Using Importance Sampling for Hybrid Bayesian Networks.* In Proceedings of SPIE Conference, Volume 5809, Orlando, 2005.

- KC Chang and Wei Sun. *Performance Modeling for Dynamic Bayesian Networks.* In Proceedings of SPIE Conference, Volume 5429, Orlando, 2004.

- KC Chang and Wei Sun. *Comparing Probabilistic Inference for Mixed Bayesian Networks.* In Proceedings of SPIE Conference, Volume 5096, Orlando, 2003.

Chapter 2: Bayesian Networks

Human exploration of Nature never stops. Unfortunately, the exact mechanisms of many natural processes are too complicated to model. For example, people won't know for sure whether or not it will rain tomorrow; meteorologist forecasts it with a probability: for example, 70% chance of rain. Born of the marriage of probability theory and graph theory, Bayesian networks (BNs) [Pearl88, Ne90], also known as belief networks, causal networks, directed Markov field, are a probabilistic modeling language at the cutting edge of artificial intelligence (AI) and statistics. Using a graphical model benefits both knowledge representation and reasoning because of the following features:

- Modularity: variables can be divided into subsets by their probabilistic dependence relationships. The qualitative structure and the quantitative parameters make the complex model to be consisted of smaller components.
- Decomposability: because of the conditional independence between variables, the joint distribution over the domain could be expressed as the product of conditional probability distributions (CPDs) according to the chain rule. This property saves computations significantly.
- Visualization: graphical models encode the independence explicitly using a visual representation. They are easy to understand and interpret and therefore

they are very appealing to non-technical users. Also the clear semantics of graphical representations make the model easy to construct.

- Inference efficiency: probabilistic inference algorithms could be developed by taking advantage of the topology of the model. They are more efficient than brute force statistical methods for general probability models.

In this chapter, we will first briefly introduce the historical evolution of Bayesian networks. Then we will present the formal definition of BN with a simple example of pure discrete BN model. After that, we will discuss the classification of various BNs and review probabilistic inference methods for different BNs. At the end of this chapter, we will present the current challenging problems in BN inference.

2.1 Brief Historical Retrospective of BN Research

The origins of Bayesian networks can be traced back as far as the early decades of the 20th century. Geneticist Sewell Wright used graphical representation and causal modeling for path analysis to aid the biometric study of genetic inheritance in 1920s [Wri21, Wri23]. His methods were formalized by economists after the Second World War [Haa43, Koo50] and adopted by social scientists in the 1970s [Gol72, Dun75]. Influence diagrams (IDs) as a formalism to model decision problems with uncertainty were introduced by Howard and Matheson in 1981 [HM81]. Influence diagrams command a unique position in the history of graphical models. On the one hand, they can be seen as an extension of path diagrams. On the other hand, because an influence diagram without decision variables and utility functions is a Bayesian network, it can also be viewed as the precursor to Bayesian networks. But since the complex representation and lack of inference efficiency, influence diagram has had only mild influence on automated reasoning (see the historical retrospection [Pearl05, HM05]).

In 1980s, the concepts of causal network, d-separation and amenable algorithms to propagate probabilistic information through the graph were introduced [Pearl82, Pearl86, Pearl88, Co84]. Around the same time, the machine-learning community developed sound methods to learn BNs from data. These developments put Bayesian networks at the forefront of AI research. Since then, much progress has been made on research about modeling and inference algorithms using Bayesian networks.

2.2 Definition and Example

There are two main types of graphical model: undirected and directed. Undirected graphical models, also known as Markov networks or Markov random fields, are more popular with the physics and vision communities. Bayesian Networks are directed graphical models in which directed arcs are used to model probabilistic dependency. Because the directed representation is natural for modeling causality, BNs are sometimes called the causal belief networks. Any ordering can be used in which knowledge of a parent node influences the probability of a child node. This influence could be logical, physical, temporal or simply conceptual in that it may be most appropriate to think of the conditional probability of a child node given its parents. We adopt the name Pearl uses, Bayesian networks, on the grounds that the name is general. However, Bayesian networks do not necessarily imply using Bayesian statistics; Indeed, it is common to use frequentists methods to estimate the parameters of the CPDs. They are so called because they use Bayes' rule for probabilistic inference.

A Bayesian network is a directed acyclic graphs (DAG) consisting of nodes and arcs where nodes represent the random variables and arcs represent the probabilistic relationships between variables. Associated with each node in a Bayesian network, a conditional probability distribution (CPD) is defined to specify the probabilistic relationship between this variable and its parents. If a node has no parents, we call such a node root node and a prior distribution for this node is specified. A joint distribution over all the random variables is determined by a fully specified Bayesian network.

The notations used in this dissertation generally follow the conventions. We denote random variables by capital letters or indexed capital letters such as A, B_i or X. The corresponding lower case letters such as a, b_i, x denote the particular instantiations of the random variables. Bold capital letters (e.g., $\mathbf{A}, \mathbf{B_i}, \mathbf{X}$) are used to denote sets of random variables and bold lower case letters such as $\mathbf{a}, \mathbf{b_i}, \mathbf{x}$ denote their realized values. We use $\mathcal{R}(A)$ to denote the set of all possible values that A can take. In this dissertation, we focus on hybrid models where both discrete and continuous random variables are involved. We usually use letters from the beginning of the alphabet (e.g., A, B, C, D) for discrete variables and letters from the end of the alphabet (e.g., W, X, Y, Z) for continuous variables. However, if we have context for the model, descriptive abbreviated letters are used for variables. The bold Greek letter $\mathbf{\Delta}$ refers to all the discrete variables and $\mathbf{\Gamma}$ refers to all the continuous variables. In conducting inference, we usually use $\mathbf{E}$ to denote the set of evidence nodes and $\mathbf{T}$ to refer the set of target nodes of our interest.

$P(X)$ will be used to denote the probability mass function of X in the discrete case or the probability density function of X in the continuous case. Some researchers use generalized probability density function [DeGr70] regardless of the types of random variables. Actually, probability mass function can be viewed as a probability density function with respect to counting measures. For a discrete random variable, $P(X = x)$ (or $P(x)$ in short) refers to the probability that X takes the value x. $L(X|\theta)$ refers to the likelihood function of X given some parameter θ. The conditional probability distribution of X given Y is denoted by $P(X|Y)$. In Bayesian networks, all nodes pointing to a particular node A are called the parents of node A. We denote the set of parents of node A by $Pa(A)$. So the conditional probability distribution (CPD) of node A is denoted by $P(A|Pa(A))$.

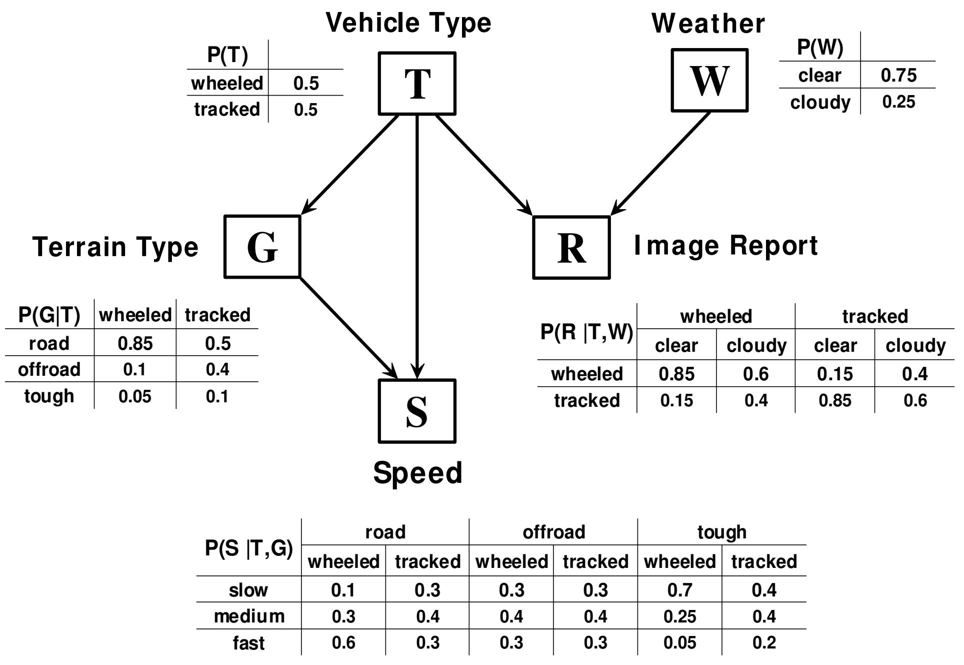
Now we are ready to give the formal definition of a Bayesian network:

Definition 2.1. *A Bayesian network $\mathcal{B}$ over the domain of the set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ is a pair $\langle \mathcal{G}, \theta \rangle$. $\mathcal{G}$ is a directed acyclic graph with each node for a random variable $X_i \in \mathbf{X}$. $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ is a set of conditional probability distributions where θ_i is the conditional probability distribution of X_i given its parents $P(X_i|Pa(X_i))$.*

The DAG $\mathcal{G}$ encodes the conditional independence of the joint probability distribution via the structure of the networks. In a BN, every node is independent of its non-descendants given its parents. According to the *probability chain rule*, the joint probability distribution induced by a Bayesian network can be decomposed as the product of CPDs for every node in the domain:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \tag{2.1}$$

This might be the most important property of Bayesian networks because it provides the theoretical basis for computationally efficient inference algorithms.



P(T)	
wheeled	0.5
tracked	0.5

P(W)	
clear	0.75
cloudy	0.25

P(G|T)	wheeled	tracked
road	0.85	0.5
offroad	0.1	0.4
tough	0.05	0.1

P(R |T,W)	wheeled		tracked	
	clear	cloudy	clear	cloudy
wheeled	0.85	0.6	0.15	0.4
tracked	0.15	0.4	0.85	0.6

P(S |T,G)	road		offroad		tough	
	wheeled	tracked	wheeled	tracked	wheeled	tracked
slow	0.1	0.3	0.3	0.3	0.7	0.4
medium	0.3	0.4	0.4	0.4	0.25	0.4
fast	0.6	0.3	0.3	0.3	0.05	0.2

Figure 2.1: A simple Bayesian network example–Vehicle Identification.

Let us take a simple vehicle identification problem as a concrete example. Suppose we want to distinguish between two types of vehicles on the ground. The vehicles of our interests are: wheeled vehicle such as truck and tracked vehicle such as tank. The weather condition affects the quality of image sent by the image sensor. And we know wheeled vehicle tends to be on the normal road, while tracked vehicle is able to be driven offroad or even on very rough terrain. Furthermore, the vehicle speed is influenced by the vehicle type and the road condition as well.

A Bayesian network to model the above problem is shown in Figure 2.1. For demonstration purpose, we discretized the originally continuous variable (Speed) to make this model a pure discrete BN. As one can see, each node in the network has a table to parameterize its dependence relationship with its parents. Logically, the terrain type on which the vehicle is being driven is highly correlated with the image report by the sensor. This is because if the image sensor reported that the vehicle is a wheeled vehicle, then we are more certain that the vehicle is on the normal road than in the case that tracked vehicle is reported. But if we are given the information that the vehicle is indeed a wheeled vehicle, then from the structure of the network, one could easily understand that terrain type is now independent of the image report. This is also consistent with our intuition. In other words, given the fact that the vehicle is wheeled, then no matter what report sent by the image sensor, it has no influence on what type of terrain this vehicle is likely to be driven on. As another example, it is clear that vehicle type is independent of weather. But if we receive from the image sensor that a wheeled vehicle is reported, then the vehicle type is now related to weather. Regarding the quantitative aspect, those tables in the network are conditional distributions of the discrete variables: for instances, $P(T = wheeled) = 0.5$, $P(R = wheeled|T = wheeled, W = cloudy) = 0.6$.

From this BN model, one can easily see the relationships between all variables involved in the problem. Prior knowledge is well modeled intuitively. Influences between various variables are easy to interpret. Of course, for complex problems and huge databases, network structure need to be learned from data as well as validated by experts. However, this simple BN model already shows the expressive power and transparent process of reasoning using BN as an excellent knowledge base. Note that without the Bayesian network model, the joint distribution over those 5 variables will be a probability table with a size of $2\times2\times2\times3\times3 = 72$, in which 71 probabilities need to be specified. But using BN model, only 22 probabilities are needed. In realistic problems where the corresponding BN model could be much larger, the savings in modeling and computations can be very significant.

Let us examine a few examples to show how to compute the joint and marginal probabilities in the vehicle identification model. Suppose we want to know the probability of a certain atomic event, e.g., $T = wheeled, W = cloudy, G = road, R = tracked, S = fast$. This can be easily computed using Equation 2.1.

$$\begin{aligned}
&P(T = wheeled, W = cloudy, G = road, R = tracked, S = fast)\\
&= P(T = wheeled)\cdot P(W = cloudy)\cdot P(G = road|T = wheeled)\cdot\\
&P(R = tracked|T = wheeled, W = cloudy)\cdot P(S = fast|T = wheeled, G = road)\\
&= 0.5 \times 0.25 \times 0.85 \times 0.4 \times 0.5\\
&= 0.02125
\end{aligned}$$

Similarly, we can compute the probability of any atomic event in this way. Furthermore, we can compute the marginal distributions of subsets of nodes by summing out the variables we are not interested in. However, summing out variables by brute force is not always computationally feasible because of the exponentially increasing size of the joint probability distributions. Furthermore, it is usually impossible to obtain closed-form results when integrating out variables if nonlinear functions or non-Gaussian distributions are involved.

In BN inference, we usually have a set of observations. The posterior distributions of the target variables of interest may change drastically given different observations.

Because the corresponding decision highly depends on the accurate posterior distributions, how to compute these distributions is one of the most important tasks using BNs. Theoretically, if we can compute the marginal distributions of subsets, we can compute the conditional probability distribution according to the definition: $P(\mathbf{X}|\mathbf{Y}) = \dfrac{P(\mathbf{X}, \mathbf{Y})}{P(\mathbf{Y})}$. Unfortunately, this is in general NP-hard [Co90, DL93]. Only for special classes of BNs, inference is tractable.

2.3 Classification of BNs

Bayesian networks can be classified into various categories from different perspectives. It is impossible to develop a general efficient inference algorithm that works for all types of BNs because of the NP-hardness of the problem. We know efficient inference algorithms are highly model-dependent. In this section, we discuss several main classes of BNs.

First, from the perspective of network structure, we have singly-connected Bayesian networks and multiply-connected Bayesian networks. Singly-connected Bayesian networks, also called polytrees, have no more than one path between any two nodes. On the other hand, multiply-connected Bayesian networks contain at least one pair of nodes that have more than one path between them. Two examples are shown in Figure 2.2. In general, multiply-connected Bayesian networks have more interactions between variables than singly-connected Bayesian networks, and inference is more computationally intensive because more variables need to be summed or integrated out. In fact, singly-connected Bayesian network with pure discrete variables or pure linear Gaussian variables, are the only special BN classes for which exact inference

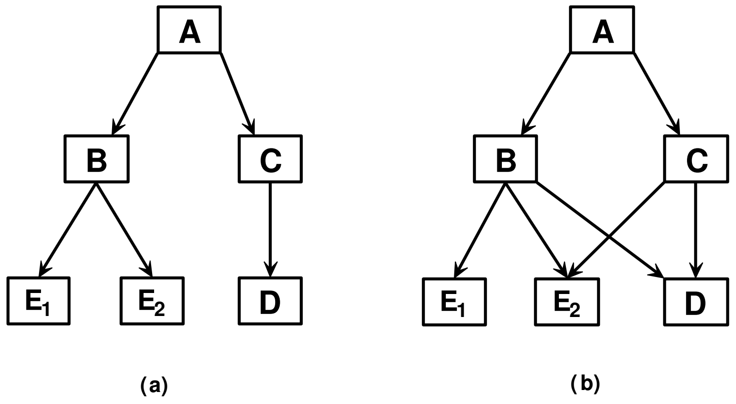


Figure 2.2: (a) A simple singly-connected Bayesian network. (b) A Multiply-connected Bayesian network. Note that two paths exist between A to D and A to E_2

can be done in linear time.

Second, according to the types of random variables in the model, there are pure discrete, pure continuous, and hybrid Bayesian networks respectively. Hybrid BNs contain both discrete and continuous variables simultaneously.

If a random variable and its parents are discrete, the CPD is usually specified as a probability table in which the entries are probabilities for values of the child

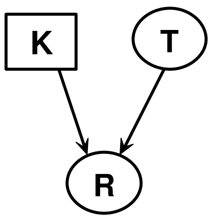


Figure 2.3: A simple 3-node hybrid model

conditional on values of its parents. But the CPD of a continuous variable usually includes a functional relationship with its parents. We show a simple hybrid model with 3 nodes in Figure 2.3. Following the convention, we will use squares or rectangles to depict discrete variables and circles or ellipses to depict continuous variables. Note that, K is a discrete variable. Without loss of generality, let us assume it has two values: true or false with probability 0.8, 0.2 respectively. Suppose the prior of node T is Gaussian with mean 30 and variance 3 denoted as $\mathcal{N}(30, 3)$. We define the CPDs of R as the following:

$$P(R|K,T) = \begin{cases} \mathcal{N}(T, 1) & K = true \\ \mathcal{N}(\sqrt{T} - 3, 9) & K = false \end{cases}$$

Note when K takes value of "false", the mean of R will be 3 subtracted from the square root of T. Through this example, we show that there could be simple linear function or complicated nonlinear function involved in CPDs for continuous variables. Also, the distributions of continuous variables are not necessarily Gaussian. The simplest hybrid model is conditional linear Gaussian model (CLG) where given all the discrete parents, the distribution of a continuous variable is a linear combination of its continuous parents with a Gaussian noise. But in general, hybrid model could be nonlinear, non-Gaussian. Since CLG is a very important hybrid model and we will use it as a basic model, here we give a formal definition of CLG:

Definition 2.2 (CLG). *A conditional linear Gaussian (CLG) is a hybrid Bayesian network containing both discrete variables (denoted as $\mathbf{\Delta}$) and continous variables (denoted as $\mathbf{\Gamma}$), with the following restrictions:*

- *A discrete node can not have any continuous parent; thus, all the CPDs for* *discrete nodes can be represented as probability tables.*

- *The CPD of any continuous variable is a linear Gaussian CPD given any combination of all of its discrete parents. More formally, if a continuous variable* Y *has discrete parents* $\mathbf{D} = \{D_1, D_2, ..., D_m\} \subseteq \mathbf{\Delta}$ *and continuous parents* $\mathbf{X} = \{X_1, X_2, ..., X_n\} \subseteq \mathbf{\Gamma}$, *and for every* $\mathbf{d} \in Dom(\mathbf{D})$, *we have* $\beta_{\mathbf{d},0}, \beta_{\mathbf{d},1}, ..., \beta_{\mathbf{d},n}$ *and* $\sigma^2_{\mathbf{d}}$, *then the CPD of* Y *is defined as:*

$$P(Y|\ \mathbf{x}, \mathbf{d}) = \mathcal{N}(Y; \beta_{\mathbf{d},0} + \sum_{i=1}^{n} \beta_{\mathbf{d},i} x_i,\ \sigma^2_{\mathbf{d}})$$

Obviously, a CLG is reduced to a multivariate Normal distribution given any assignment of all discrete variables in the model. It follows that the joint distribution represented by a CLG is a Gaussian mixture where each Gaussian component corresponds to an instantiation of all discrete variables.

This dissertation focuses on a special class of hybrid BN models in which no discrete node has any continuous parent. We call this special hybrid BN conditional hybrid model (CHM).

Definition 2.3 (Conditional Hybrid Model)**.** *A conditional hybrid model (CHM) is a hybrid Bayesian network containing both discrete variables and continuous variables with the only restriction that no discrete variable may have a continuous parent.*

Obviously, the CLG is a special case of the CHM. In CHMs, we may have nonlinear functional relationships or non-Gaussian variables.

Third, from the process point of view, static Bayesian networks could be extended to dynamic Bayesian networks (DBNs) to model stochastic process. Partially dynamic

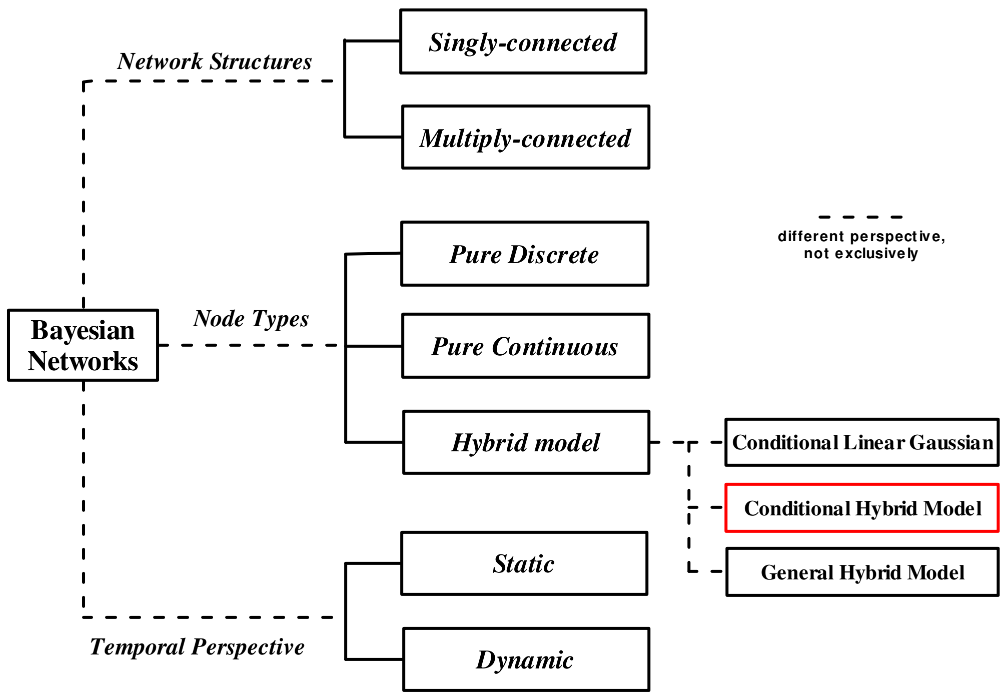


Figure 2.4: Taxonomy for Bayesian networks. This dissertation focuses on inference for conditional hybrid model.

Bayesian network (PDBN) is a special class of DBN, also called temporal Bayesian network, in which some of the nodes do not change their values over time (but their beliefs do change). Many applications such as systems diagnosis, target tracking, vision and speech recognition use PDBN models. Also, DBNs generalize hidden Markov model (HMM) and state space model; for details, see [Mur02].

Furthermore, it is natural to have any combination of the above mentioned Bayesian networks to model complicated situations. It is well known that one of the most difficult BN models is the hybrid nonlinear non-Gaussian dynamic Bayesian network.

As a summary, a taxonomy tree for various Bayesian networks is shown in Figure 2.4. Please note that these different perspectives are not exclusive and they could be combined to formulate complicated BN models.

2.4 Probabilistic Inference using BNs

One of the main purposes of constructing Bayesian networks is to perform probabilistic inference. A typical inference problem is to compute the posterior probability distribution of a set of query variables of interest given observations for a set of evidence nodes. For example, in the vehicle identification model shown in Figure 2.1, if we observed that the vehicle was fast on the road and received an image report by the sensor showing that the vehicle was wheeled, we would like to know the probability that it is indeed a wheeled vehicle. This is equivalent to computing the posterior probability $P(T = wheeled|S = fast, R = wheeled)$. Since the vehicle identification model is a simple pure discrete BN, exact inference can be done. It turns out that the posterior probability of being a wheeled vehicle given the above observations is 87.4%, while its prior probability is 50%.

Over the last several decades, a variety of inference algorithms has been proposed in the literature. Unfortunately, it has been proven that in general both exact and approximate inference for Bayesian networks are NP–hard [Co90, DL93]. Formally, given a Bayesian network $\mathcal{B}$ over the variables $\mathbf{X}$, we have some query variables $\mathbf{Q}$ and a set of evidence variables $\mathbf{E}$ where $\mathbf{Q}, \ \mathbf{E} \subseteq \mathbf{X}$. Typically we need to compute the posterior probability distribution $P(\mathbf{Q} \mid \mathbf{E} = \mathbf{e})$. But even in the simplest case when $\mathbf{E} = \emptyset$, the inference is still NP–hard.

Theorem 2.1. *Given a Bayesian network $\mathcal{B}$ over variables $\mathbf{X}$ and some variables*

$\mathbf{Q} \subseteq \mathbf{X}$*, then in general computing* $P(Q)$ *is NP–hard, even if* $|\mathbf{Q}| = 1$.

This theorem was proven by Cooper in [Co90]. Furthermore, the seemly easier approximate inference in Bayesian networks is NP–hard too. The following theorem was proven by Dagum and Luby in [DL93].

Theorem 2.2. *Given a Bayesian network* $\mathcal{B}$ *over binary discrete variables* $\mathbf{X}$ *and some variables* $\mathbf{A}$ *and* $\mathbf{E}$ *where* $\mathbf{A},\ \mathbf{E} \subseteq \mathbf{X}$*, then unless* $NP \subseteq P$*, there does not exist any polynomial-time approximate inference algorithm to compute* $P(\mathbf{A} \mid \mathbf{E} = \mathbf{e})$ *with absolute error less than 0.5, even if* $|\mathbf{A}| = 1$ *and* $|\mathbf{E}| = 1$.

Also, although exact inference in discrete polytree Bayesian networks can be done in linear time, Lerner proved in his PhD dissertation [Lern02] that both exact and approximate inference in polytree CLG models are NP–hard.

Theorem 2.3. *Given a polytree CLG* $\mathcal{T}$ *with some binary discrete variables* $\mathbf{A}$ *and some evidence set* $\mathbf{E}$*, then computing* $P(\mathbf{A} \mid \mathbf{E} = \mathbf{e})$ *is NP–hard. Furthermore, unless* $NP \subseteq P$*, there does not exist any polynomial-time approximate inference algorithm with absolute error less than 0.5.*

These hardness results might make one conclude that probabilistic inference using Bayesian networks is a lost game. Fortunately, this is not true. The correct implication from the proven NP-hardness is that developing an efficient inference algorithm applied to all classes of Bayesian networks seems unlikely. Knowing the NP-hardness suggests us to avoid searching for a general algorithm and we should focus on inference for special classes of Bayesian networks which have special structure or features we can exploit to design efficient algorithms. The real-life problems we deal with are usually simpler than the ones used in NP-hardness proofs. And of course, we could deliberately model the problem into the special classes for which we can conduct inference in an efficient way. In fact, researchers have proposed a number of inference algorithms for various classes of Bayesian networks in the literature. In the remainder of this section, we will review these methods for exact and approximate inference and elucidate the relationships between them.

2.4.1 Exact Inference

The first exact inference algorithm is Pearl's belief propagation, also known as Pearl's message passing, presented in early 1980s; details of the algorithm are well summarized in [Pearl88]. For polytree pure discrete or pure linear Gaussian BN, Pearl's message passing algorithm guarantees to return the correct marginal distribution for every hidden node in a finite number of iterations of message propagation between all variables. The algorithm can be implemented in a centralized fashion in which case it converges in two iterations [PS91]. It is also amenable to parallel updating. Unfortunately, Pearl's message passing algorithm was designed originally for polytree Bayesian networks and exact results can be obtained only for this particular network class.

Pearl also proposed an exact inference method for multiply connected networks, called loop cutset conditioning [Pearl86]. A selected subset referred as cutset is instantiated, and then the original network is changed to be singly connected by conditioning. Inference solutions of these singly connected networks are combined by weighing the prior probabilities of the cutset. The cutset conditioning algorithm can provide exact inference but its complexity grows exponentially with the size of the cutset.

The most popular exact inference algorithm for multiply connected discrete Bayesian networks is clique tree algorithm introduced by Lauritzen and Spiegelhalter [LS88, SS90]. It is also called Junction tree or clustering algorithm. Clique tree algorithm transforms the original multiply-connected network into an undirected singly connected graph, called clique tree. First, an underlying undirected graph is obtained by replacing directed edges in Bayesian networks with undirected edges. Next, parents of the common node are connected pairwisely. Connecting, or marrying, the parents leads the name of this process moralization. And then, loops in the undirected graph that have 4 or more nodes are broken up into loops containing at most 3 nodes. In other word, some chords are inserted in the loops and it makes the undirected graph chordal. A descriptive name of this process is called triangulation. How to triangulate the undirected graph aims to induce small cliques. Cliques are then formulated as factors over the related variables from the moralized, triangulated undirected graph. The clique tree built from a Bayesian networks is not necessarily unique. Finding the optimal clique tree, which has the smallest maximal clique, is NP–hard. This is not surprising because finding the optimal variable elimination order is NP–hard.

A clique tree is an undirected tree consisting of nodes and edges. Each node in a clique tree is a clique or a cluster. Each clique is associated with a factor, called potential, over the variables the clique contains. Each edge is associated with a factor, called sepset, over the intersection of the variables of two cliques it connects. After building the corresponding clique tree from a Bayesian network, clique tree algorithm performs message propagation between cliques. Each clique sends a message to its neighbors after receiving messages from its other neighbors. After a total of $2n-2$

messages passing if we have n cliques connected by $n-1$ edges, then each potential and each sepset in the clique tree converges to the correct marginal distribution over its variables respectively.

There are several algorithms closely related to the clique tree algorithm, such as *arc reversal* by Shachter [Sh86,Sh90], *variable elimination algorithm* [ZP94], and *symbolic probabilistic inference* (SPI) by Shachter et al. [SD90]. Arc reversal algorithm applies a sequence of operations using Bayes' Rule to reverse the links. The process continues till network is reduced to only the query nodes and evidence nodes as directed predecessors. Variable elimination algorithm eliminates other variables one by one by summing them out. An optimal elimination ordering results in the least computational complexity. However, it is NP-hard, as mentioned earlier. SPI formulates the probabilistic inference problem as a combinatorial optimization problem and solves it by finding the optimal factoring.

For pure continuous model, if all variables are Gaussian and the relationship between every variable and its parents is linear, then the joint distribution represented by the Bayesian networks is a multivariate Gaussian. Therefore all inference can be done using this joint Gaussian distribution. Alternatively, we can use clique tree algorithm with special form of representations of factors for potentials and sepset over Gaussian variables.

Regarding hybrid models, relatively little has been done so far. The simplest hybrid model is conditional linear Gaussians (CLGs) and it is a hybrid model for which exact inference could be done. In general, even approximate inference for polytree CLGs is NP–hard. The state of art algorithm for exact inference in CLGs is Lauritzen's algorithm [Lau92, LJ01]. Lauritzen's algorithm is based on the clique tree algorithm [LS88,SS90,HD96] originally developed for discrete Bayesian networks. Lauritzen's algorithm returns the exact answer in the sense that the first and second moments of the posterior distribution are correct, while the true distribution might be a mixture of Gaussians. However, a workable clique tree for Lauritzen's algorithm may be exponentially large, making the algorithm intractable.

In general the complexity of all exact inference algorithms is exponential in the size of the largest clique of the triangulated moral graph, which is also called the induced width of the graph [LS88]. For networks with many loops or general hybrid models, intractability rules out the use of any exact inference algorithm. Therefore, approximate inference methods come to the stage.

2.4.2 Approximate Inference

The major approximate inference methods for Bayesian networks include model simplification, stochastic sampling and loopy belief propagation.

Model simplification methods first simplify the model until exact methods become feasible and then apply an exact algorithm. Some commonly applied simplification methods include removal of weak dependency or arc removal, discretization of continuous nodes, linearization of nonlinear relationships, and state space abstraction.

Stochastic sampling, also called Monte Carlo simulation, is the most popular method for approximate inference and it has been used extensively for probabilistic inference. In general, this method is applicable to every class of Bayesian networks except some specialized sampling methods that may use techniques limited to particular types of BNs. However, sampling algorithms may take a long time to converge to reliable answers. Especially with very unlikely evidence, sampling algorithms may not converge even with huge sample sizes. Basically, stochastic sampling algorithms first generate a set of random samples or instantiation of the network according to some pre-selected distributions, and then approximate the posterior distributions of the query nodes by the frequencies of appearances in the samples. The accuracy of the inference depends on the sample size and the selected distributions used for sampling, but it can be irrespective of the structure of the networks and the CPDs of variables. Stochastic sampling can be further divided into two categories: importance sampling algorithms and Markov Chain Monte Carlo methods (MCMC) [GRS96].

The first sampling algorithm for Bayesian network was proposed by Henrion in 1988 [Hen88], and is called logic sampling. Logic sampling uses simple forward sampling according to the prior distribution of the network and simply discards the samples not consistent with the evidence. It is very inefficient and performs poorly when evidence is unlikely. Logic sampling is not feasible when we have evidence for continuous variables because the probability of generating the same continuous value is zero. As an improved version, the likelihood weighting (LW) [FC89, SP90] method was designed to overcome the problems of logic sampling. LW does not sample the nodes already observed. Instead, it takes the observed value of evidence node and weights the sample by the likelihood of evidence conditional on the sample. LW performs significantly better than logic sampling and can handle very large, complicated networks. But it still converges slowly for unlikely evidence since LW uses the prior distribution to generate samples as well. When evidence is very unlikely, LW may hardly have the representative samples of the unknown posterior distributions given the evidence because the weight of random sample is very small.

Both logic sampling and LW use the prior distribution of the network to generate samples. This is the reason why it performs poorly with unlikely evidence because the true distribution given unlikely evidence could be far away from the prior distributions. It is well-known that the performance of sampling methods depends not only on the sample size, but more so on the sampling distribution. Instead of using the prior, another idea originated from the finite-dimensional integral, is to use an "importance" function for sampling. This leads to the concept of importance sampling. In importance sampling, an importance function is a known probability distribution we could use for sampling. In principle, the closer the importance function to the true unknown distribution, the more efficient and accurate the sampling algorithm will be.

The state-of-the-art importance sampling algorithm to deal with unlikely evidence for purely discrete networks is *Adaptive Importance Sampling for Bayesian Networks* (AIS-BN) proposed by Jian Cheng and Marek Druzdzel [CD00]. In AIS-BN, they proposed a concept of *Importance Conditional Probability Table* (ICPT) defined as the conditional probability table for every node given its parents as well as evidence. And based on the assumption that the network structure does not change after absorbing the evidence, AIS-BN uses the product of ICPTs as the importance function for sampling. ICPT for every node could be learned during the sampling and the importance function is updated accordingly. Their experiments show good results for very large networks with extremely unlikely evidence.

Based on AIS-BN, another method called *Evidence Pre-propagation Importance Sampling Algorithm* (EPIS-BN) was proposed in 2003 [YD03]. In EPIS-BN, the learning stage of ICPTs is avoided because it uses loopy propagation to compute ICPTs directly. Their experimental results show that EPIS-BN saves computations and time without losing performance compared with AIS-BN.

All of the above mentioned sampling methods are under the framework of importance sampling, in which samples are independently generated. They only differ in how they choose and update the importance function and how they generate and weigh the sample. Logic sampling and likelihood weighting choose the prior distribution as the importance function and never update the function. AIS-BN and EPIS-BN choose and update the importance function by learning or loopy propagation and they perform very well even with extremely unlikely evidence. Unfortunately, AIS-BN and EPIS-BN only work for pure discrete networks.

Importance sampling algorithms generate independent random samples for approximate inference. But another stochastic sampling approach, called Markov Chain Monte Carlo (MCMC), generates dependent random samples for approximate inference. In particular, MCMC uses Markov Chain to generate the samples. The stationary distribution of the Markov Chain is the target distribution from which we want to sample. MCMC usually needs certain amount of time before generating useful samples. This is called burning time. Two popular MCMC methods are Metropolis-Hasting sampling [MRR+53, Hast70] and Gibbs sampling [GG84]. Pearl proposed an approximate inference method for BNs using Gibbs sampling in [Pearl87].

The third main category of approximate inference for BNs is loopy belief propagation (LBP). Recall that Pearl's message passing works exactly for polytree discrete BN. When loops (undirected cycles) are present in the network (multiply-connected BN), local propagation may run into problem due to the non-unique path. As Pearl noted, the messages propagated between variables in the loops may not be correct:

When loops are present, the network is no longer singly connected, and

> local propagaion schemes will invariably run into trouble. The reason is both architectural and semantic. If we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around these loops, and the process may not converge to a stable equilibrium. . . . Such oscillations do not normally occur in probabilistic networks because of the stochastic nature of the link matrices, which tend to bring all messages toward some stable equilibrium as time goes on. However, this asymptotic equilibrium is not coherent, in the sense that it does not represent the posterior probabilities of all nodes of the network. The reason for this is simple: all of our propagation equations were based on some conditional independence assumptions that might be violated in multiply connected networks. [Pearl88, p.195]

However, researchers still use this algorithm as an approximate inference method for multiply connect BNs. This extended method is so-called loopy belief propagation (LBP). In recent years, loopy belief propagation – applying Pearl's message passing algorithm for the networks with loops – has become a popular topic in the literature [MWJ99,WF99] because of its simplicity of implementation and its good performance. Researchers have found that loopy belief propagation usually converges, and when it converges, it provides good estimation empirically. In this dissertation, we propose two approximate inference algorithms under the framework of message passing to deal with nonlinear and non-Gaussian variables in hybrid BN models.

2.4.3 Mixed Inference

For complicated networks, under some circumstances such as conditioning or density approximation, we can use exact inference for at least part of the networks. In these cases, a general approach combining both exact and approximate inference may be the most appropriate method. We call this approach mixed inference. Mixed inference may achieve the best of both worlds.

Recall that the popular clique tree algorithm works for discrete networks and its extended version Lauritzen's algorithm works for the simplest hybrid model CLG. For arbitrary hybrid Bayesian networks, it is possible to use the same algorithmic framework if we can represent the clique potentials by appropriate distributions. [KL99] proposed a generalized clique tree algorithm for hybrid BNs. They used approximate inference, such as importance sampling, to estimate the densities in each clique and messages sent between cliques. Their approach is a general schema which can be instantiated in different ways based on the representation formats and different ways to manipulate.

Theoretically, we know Gaussian mixture can approximate any continuous distribution at arbitrary accuracy with sufficient number of Gaussian components. [Poland94, Shenoy06] proposed using finite mixture of Gaussians to fit arbitrary continuous distributions in hybrid BNs. They then conduct exact inference using Lauritzen's algorithm.

Similarly, researchers proposed that mixture of truncated exponentials (MTE) can approximate any probability density function and it is very useful for inference in hybrid BNs [MRS01, MRS02, CS06]. MTE can always be marginalized in closed-form. Therefore, it allows message propagation can be done exactly using the same architecture of clique tree algorithm.

Another instantiation of mixed inference algorithm is called cutset sampling [BD06], which combines sampling method and exact polytree algorithm. Cutset sampling can be viewed as the an anytime approximation of exact cutset conditioning algorithm [Pearl86]. It applies Gibbs sampling [Neal93] on a selected loop cutset and performs exact inference on the rest of the network. Their experimental results show that cutset sampling outperform AIS-BN with unlikely evidence. However, it only works for discrete Bayesian networks.

Mixed inference takes advantage of the best features from both approximate and exact inference. Like approximate method, it can deal with complex domain, including different type of variables such as sampling methods; while exact inference can exploit the locality structure of BNs to reduce the dimensionality of the probability densities such as clique tree algorithm.

2.4.4 Sequential Inference for Dynamic Bayesian Networks

When some or all random variables in Bayesian networks change their values over time, it becomes dynamic Bayesian network (DBN). Sequential inference is the most popular framework for DBNs because it is not feasible to unroll the network for all time slices during the dynamic process. Sequential inference is to estimate the posterior distribution recursively when next observations become available based on the estimation at the current time. Sequential inference could be exact in special cases: if we can obtain the exact posterior distribution for the unrolled two-time-step networks after absorbing evidence at the current time slice, then we can do exact rollup for the next time slice for dynamic Bayesian networks [TAW02]. As an example, *Kalman Filter* provides exact answers for linear Gaussian model [Kal60, AM79, BLK01].

In the past few years, a new filtering method called *Unscented Kalman Fitler* (UKF) has been developed to deal with nonlinear dynamic transition for Gaussian model [JU96, JU97, Julier02]. UKF uses deterministic sampling method and can perform estimation with accuracy up to the second moment of the continuous distribution. It is one of the best performing sequential inference algorithms to date for nonlinear Gaussian models.

In general, we have to approximate the posterior distribution to make inference tractable. This is because the belief state will become fully correlated only after a few time slices in dynamic Bayesian networks. Boyen-Koller's method provides a general approximate rollup framework for sequential inference [BK98]. Boyen-Koller's method proved that the estimation error resulting from the approximation remains bounded because of the stochasticity of the process. However, it runs into trouble when the process is close to deterministic (error bound will be meaninglessly too big in this case).

Particle filter (PF), also known as sequential Monte Carlo simulation, is a general method of sequential inference [DFG01a, DFG01b, AMG02]. In particle filtering, random samples called particles are generated and propagated through the dynamic process. Then the particles survived after resampling are taken as the samples to estimate the posterior distribution. Theoretically, if we use infinite number of particles, PF will converge to the exact answer. But in practice, particle filter often encounter the problems of particle degeneracy. Particle degeneracy means that only few paricles have significant weights, whereas the majority of particles are weighted close to zero.

This causes that the particles are not representative for a distribution. To overcome this weakness, there are many variants of particle filter proposed in the literature such as auxiliary particle filter [PS99], unscented particle filter [MDFW00], and particle filter with move by MCMC [GB01]. They differ in how they choose the proposal function for sampling and how they resample the particles. Among them, unscented particle filter (UPF) combines the methods of unscented kalman filter and particle filter. It uses the estimated distribution by unscented kalman filter as the proposal sampling distribution, and then generate the random samples correspondingly. Their experimental results show that UPF performs very well in tracking, option pricing.

2.4.5 Summary of Inference Algorithms

Inference methods proposed in the literature differ in many ways. Here are some of the main aspects:

- What topologies can it handle?
- What node types can it handle?
- Does it provide exact or approximate inference?

The NP-hardness of probabilistic inference for Bayesian networks suggests that it is not likely to find a general inference algorithm works for all situations. Indeed, the efficient methods are highly model-dependent. The best approach is to take advantages of the special features of a particular model and seek the efficient inference methods accordingly.

2.5 Challenges of BN Inference

Since the introduction of Bayesian networks in 1980s, a great deal of research work has focused on inference problems. But the overwhelming majority of them consider the case of pure discrete Bayesian networks. Both exact and approximate inference algorithms for discrete Bayesian networks are now very well understood and developed. However, many real-world problems have continuous attributes as well. Unfortunately, at the current edge, very little is done for general hybrid models. Only for the simplest hybrid model-CLG, exact inference may be obtained but not guaranteed by Lauritzen's algorithm. Generally speaking, no exact inference is possible for nonlinear, non-Gaussian hybrid Bayesian networks.

In modern applications, the size of realistic BN models is becoming larger and larger. Network structures may be of any complicated topology. Different types of variables may be mixed together in a model and their distributions could be arbitrary. To address these challenges, this dissertation presents an approximate inference framework for efficient computation and sampling.

As mentioned earlier, sampling algorithm is an ultimate alternative when all other approaches fail. This is because of its special features:

- Sampling algorithms can be model-free. The performance of sampling methods is generally improved by increasing the sample size, irrespective to the topology and types of nodes of the model. The complexity is linear in the number of samples.

- Sampling algorithms are any-time methods. That is, they can return results at any time. This is very important in time-critical applications.

- Sampling algorithms can converge very quickly if a good proposal function is used for sampling.

AIS-BN [CD00], EPIS-BN [YD03], and cutset sampling [BD06] are the most efficient sampling algorithms for pure discrete networks reported so far. AIS-BN and EPIS-BN can deal with extremely unlikely evidence by learning or computing a good importance function. However, for a continuous node in a hybrid model, given its discrete parents, a functional relationship between a continuous node and its continuous parents may be involved. How to learn or estimate the relationship will be the key for sampling in a hybrid model. We did some related studies in our paper [SC05], in which we proposed a way to approximate general nonlinear relationships by linear functions.

As well-known, the performance of sampling algorithms depends on the sample size and sampling distribution. In practice, how to find a good importance function is key. It could reduce the sample size and thereby save significantly on computation and time.

In this dissertation, we focus on another direction inspired by the traditional message passing [Pearl88]. As mentioned earlier in this chapter, loopy belief propagation has been very popular for approximate inference in recent years. In this algorithm, probabilities and likelihood encoded in messages are propagated between variables. Every variable in a BN model sends messages to its neighbors, while receiving messages from its parents and children in the mean time. After a finite number of iterations, message passing usually converges. If it converges, empirical experience indicates that the marginal distributions computed by the final messages are very close to the true marginal distributions. However, to apply message passing in a hybrid BN, we first need to address two important issues: (1) How to represent and propagate messages for continuous variables; and (2) How to integrate/combine messages between different types of variables. We will discuss these issues in the next several chapters.

Chapter 3: Unscented Message Passing

3.1 Introduction

Pearl's message passing algorithm [Pearl88] is the first exact inference algorithm for Bayesian networks. It provides correct inference results for discrete BNs with polytree network structure. Applying Pearl's algorithm to a network with loops provides approximate answers. This method is called loopy belief propagation. Due to its simplicity of implementation and its good performance, loopy propagation has become very popular in recent years [MWJ99, WF99]. In discrete case, messages are represented and manipulated by probability vectors and conditional probability tables (CPTs), which is relatively straightforward mathematically and algorithmically. But when applying Pearl's algorithm for continuous variables, it is more complicated to represent and manipulate the messages. In this chapter, we present an approximate algorithm to extend Pearl's message passing for handling continuous variables. First, we propose to use the first two moments, the mean and variance of a probability distribution, to represent message for a continuous distribution. When the distribution is Gaussian, the mean and variance is sufficient to characterize the distribution. However, if the distribution is arbitrary continuous distribution, theoretically we can use Gaussian mixture to approximate it at any accuracy with enough number of Gaussian components. Secondly, we propose to integrate messages by weighting them according to the inverse of their uncertainty (variance). The computational manipulations in message propagation equations include product of messages and multiplication of messages with the conditional probability distribution that also represented approximately by the mean and variance in continuous case. The product of messages is essentially the data fusion for multiple estimates. And the computations involving CPDs may need to consider functional transformations because the specification of a CPD for a continuous variable may include function. To deal with the potentially nonlinear functional relationship between continuous variables, we propose to use the unscented transformation [JU96, Julier02] to derive the corresponding messages. The unscented transformation uses a deterministic sampling scheme and can provide good approximations of the first two moments for a continuous variable subjected to a nonlinear transformation.

In this novel algorithm, unscented transformation plays a key role for computing continuous messages. This is why we call the new developed algorithm 'Unscented Message Passing' (UMP) [SC07a]. This chapter describes how UMP computes and propagates messages for arbitrary continuous variables. We will first review Pearl's message passing algorithm, and then show how UMP can extend this algorithm in the continuous case.

3.1.1 Pearl's Message Passing Review

Pearl's message passing, also called Belief propagation, was originally developed for polytree discrete BN. In a polytree network, any node X partitions the network into two separate parts $\mathcal{U}_X$ and $\mathcal{D}_X$, where $\mathcal{U}_X$ denotes the network "above" X and $\mathcal{D}_X$ denotes the network "below" X. One important property by this partition is X d-separates variables in $\mathcal{U}_X$ from variables in $\mathcal{D}_X$. Similarly, a link $T \rightarrow X$ also divides the network into two parts: $\mathcal{U}_{TX}$ and $\mathcal{D}_{TX}$. It is easy to see that erasing the arc $T \rightarrow X$ breaks the network into two entirely disjoint sub-networks. $\mathcal{U}_{TX}$ is the sub-network from which the arc $T \rightarrow X$ emanates and $\mathcal{D}_{TX}$ is the one to which the arc $T \rightarrow X$ points. Figure 3.1 (a) illustrates those sub-networks. Obviously,

$$\mathcal{U}_{TX} = \mathcal{U}_T + (\mathcal{D}_T - \mathcal{D}_{TX})$$

$$\mathcal{D}_{TX} = \mathcal{D}_X + (\mathcal{U}_X - \mathcal{U}_{TX})$$

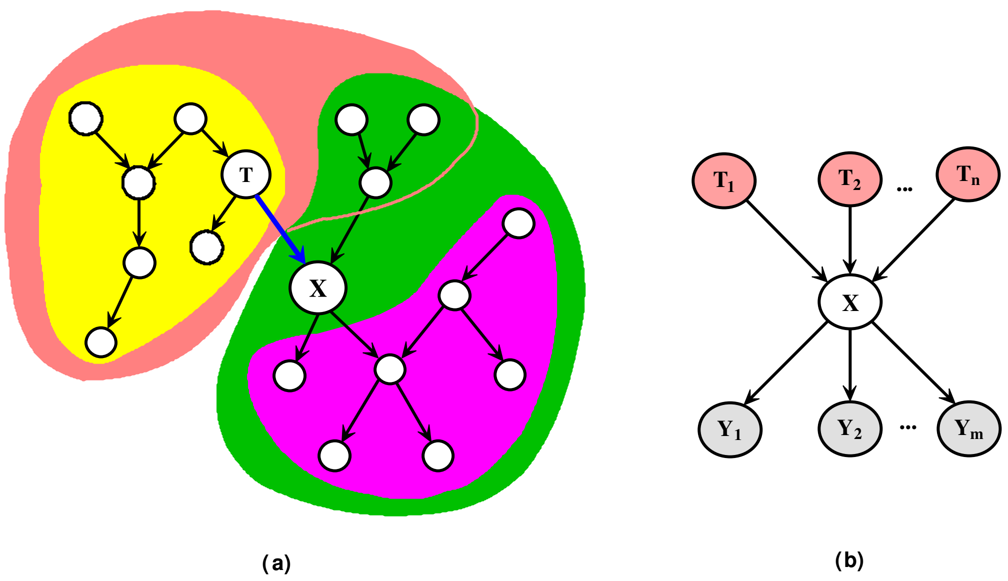


Figure 3.1: (a) 4 sub-networks divided by node X and arc $T \rightarrow X$. Orange is $\mathcal{U}_X$; yellow is $\mathcal{U}_{TX}$; green is $\mathcal{D}_{TX}$; pink is $\mathcal{D}_X$. (b) A typical node X with m parents and n children in a polytree.

Assume we have evidence observed as $\mathbf{e}$. For a node X in the polytree, we could divide $\mathbf{e}$ into two separate sets $\mathbf{e}_X^+, \mathbf{e}_X^-$, where $\mathbf{e}_X^+$ represents the observations from $\mathcal{U}_X$ and $\mathbf{e}_X^-$ stands for observations from $\mathcal{D}_X$. Also, given an arc $T \rightarrow X$, evidence $\mathbf{e}$ can be divided into $\mathbf{e}^+_{TX}$ and $\mathbf{e}^-_{TX}$. Here, $\mathbf{e}^+_{TX}$ denotes the observations from the sub-network on the tail side of the link $T \rightarrow X$ ($\mathcal{U}_{TX}$) and $\mathbf{e}^-_{TX}$ denotes the observations from the sub-network on the head side of the link $T \rightarrow X$ ($\mathcal{D}_{TX}$). Note $\mathbf{e}^-_X \subseteq \mathbf{e}^-_{TX} \subseteq \mathbf{e}^-_T$, $\mathbf{e}^+_T \subseteq \mathbf{e}^+_{TX} \subseteq \mathbf{e}^+_X$.

In Pearl's message passing algorithm, each node maintains two values called the λ value and the π value. The λ of node X is $\lambda(X) = P(\mathbf{e}^-_X|X)$, which is the likelihood of observations $\mathbf{e}^-_X$ given X. The π of node X is $\pi(X) = P(X|\ \mathbf{e}^+_X)$, which is the conditional probability of X given $\mathbf{e}^+_X$. Therefore, the belief of X (posterior distribution of X given evidence) denoted by $BEL(X)$ is

$$BEL(X) \equiv P(X|\ \mathbf{e}^+_X, \mathbf{e}^-_X) = \alpha P(\mathbf{e}^-_X|X, \mathbf{e}^+_X)P(X|\ \mathbf{e}^+_X) = \alpha P(X|\ \mathbf{e}^+_X)P(\mathbf{e}^-_X|X)$$

$$= \alpha\pi(X)\lambda(X)$$

where α is a normalization constant. The third equality follows from the fact that X d-separates $\mathbf{e}^-_X$ from $\mathbf{e}^+_X$. Usually, we do not know $\pi(X), \lambda(X)$ directly unless $\mathbf{e}^+_X$ is the only parent of X and X is the only parent of $\mathbf{e}^-_X$.

Let us choose a typical node X with m parents and n children illustrated in Figure 3.1 (b), where $T_1, T_2, ..., T_m$ denote m parents of X and $Y_1, Y_2, ..., Y_n$ denote n children of X. We will show that $\pi(X)$ and $\lambda(X)$ could be computed using π and λ messages sent from its parents and children.

It is easy to see that $\mathbf{e}^+_X$ and $\mathbf{e}^-_X$ can be further decomposed as the following:

$$\mathbf{e}^+_X = \{\mathbf{e}^+_{T_1X}, \mathbf{e}^+_{T_2X}, ..., \mathbf{e}^+_{T_mX}\}, \qquad \mathbf{e}^-_X = \{\mathbf{e}^-_{XY_1}, \mathbf{e}^-_{XY_2}, ..., \mathbf{e}^-_{XY_n}\}$$

Then

$$
\begin{aligned}
\lambda(X) &\equiv P(\mathbf{e}_X^-|X) \\
&= P(\mathbf{e}_{XY_1}^-, \mathbf{e}_{XY_2}^-, ..., \mathbf{e}_{XY_n}^-|X) \\
&= P(\mathbf{e}_{XY_1}^-|X) \cdot P(\mathbf{e}_{XY_2}^-|X) \cdots P(\mathbf{e}_{XY_n}^-|X) \\
&= \prod_{j=1}^{n} \lambda_{Y_j}(X) \qquad (3.1)
\end{aligned}
$$

where

$$\lambda_{Y_j}(X) = P(\mathbf{e}_{XY_j}^-|X) \qquad (3.2)$$

The quantity $\lambda_{Y_j}(X)$ is the λ message sent to X from its child Y_j. Also

$$
\begin{aligned}
\pi(X) &\equiv P(X|\ \mathbf{e}_X^+) \\
&= \sum_{T_1,T_2,...,T_m} P(X|\ T_1, T_2, ..., T_m)P(T_1, T_2, ..., T_m|\ \mathbf{e}_{T_1X}^+, \mathbf{e}_{T_2X}^+, ..., \mathbf{e}_{T_mX}^+) \\
&= \sum_{T_1,T_2,...,T_m} P(X|T_1, T_2, ..., T_m)P(T_1|\ \mathbf{e}_{T_1X}^+) \cdot P(T_2|\ \mathbf{e}_{T_2X}^+) \cdots P(T_m|\ \mathbf{e}_{T_mX}^+) \\
&= \sum_{\mathbf{T}} P(X|\ \mathbf{T}) \prod_{i=1}^{m} \pi_X(T_i) \qquad (3.3)
\end{aligned}
$$

where $\mathbf{T} = \{T_1, T_2, ..., T_m\}$ and

$$\pi_X(T_i) = P(T_i|\ \mathbf{e}_{T_iX}^+) \qquad (3.4)$$

The quantity $\pi_X(T_i)$ is the π message sent to X from its parent T_i.

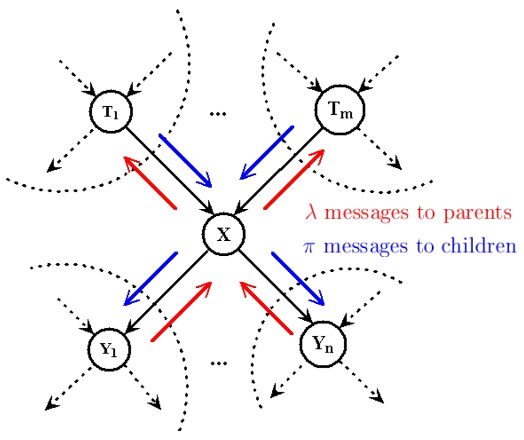


Figure 3.2: A typical node X with m parents and n children.

From Equations 3.1 to 3.4, we see that computing λ value of one node requires the corresponding λ messages sent from all of its children. Furthermore, computing the π value of one node needs the corresponding π messages sent from all of its parents. Essentially, λ message is propagated from child to parent; while π message is passed from parent to child. The propagating messages for a typical node X in a polytree network are illustrated in Figure 3.2, where $\mathbf{T}(T_1, T_2, ..., T_m)$ are the m parents of X and $\mathbf{Y}(Y_1, Y_2, ..., Y_n)$ are X's n children.

As a summary, the conventional propagation equations of Pearl's message passing algorithm are the following [Pearl88, p183] (For detailed derivation, see [Pearl88]):

$$BEL(X) = \alpha\pi(X)\lambda(X) \tag{3.5}$$

$$\lambda(X) = \prod_{j=1}^{n} \lambda_{Y_j}(X) \tag{3.6}$$

$$\pi(X) = \sum_{\mathbf{T}} P(X|\ \mathbf{T}) \prod_{i=1}^{m} \pi_X(T_i) \tag{3.7}$$

$$\lambda_X(T_i) = \sum_X \lambda(X) \sum_{T_k:\ k\neq i} P(X|\mathbf{T}) \prod_{k\neq i} \pi_X(T_k) \tag{3.8}$$

$$\pi_{Y_j}(X) = \alpha \left[\prod_{k\neq j} \lambda_{Y_k}(X)\right] \pi(X) \tag{3.9}$$

where $\lambda_{Y_j}(X)$ is the λ message sent to node X from its child Y_j, $\lambda_X(T_i)$ is the λ message sent to its parent T_i from node X; $\pi_X(T_i)$ is the π message sent to node X from its parent T_i, $\pi_{Y_j}(X)$ is the π message sent to its child Y_j from node X; and α is a normalization constant.

When this algorithm is applied to polytree discrete network, the algorithm converges in one upward and one downward pass. And the marginal belief of every hidden node is equal to the true posterior probability distribution given evidence. For a network with loops, we can still apply this algorithm. This is the so-called "loopy propagation". In general, loopy propagation will not provide exact inference results. But empirical investigations on its performance have reported surprisingly good results.

For discrete variables, messages could be represented by probability vectors and CPTs, which are conditional probability tables with finite entries. Therefore the calculations in the above formulae involve product of vectors and multiplication of vector and matrices. These are straightforward to implement. However, for continuous variables, message representation and the corresponding calculations are much more complicated. First, integrals replaces the summations in the above equations. Furthermore, since a continuous variable could have an arbitrary distribution over a continuous space, it is usually not possible to obtain closed-form analytical results when multiplying different continuous distributions together. In order to make the computations feasible while keeping the key information, we use the first two moments, the mean and variance of a continuous distribution, to represent message for a continuous variable regardless of the original distribution. Then, by assuming they are Gaussians, the multiplications of different continuous distributions could be approximated similar to fusing multiple estimates. Note that in the continuous case, $P(X|\ \mathbf{T})$ is a continuous conditional probability distribution and it may encode arbitrary function of continuous variables. To integrate the product of continuous distributions as shown in Equations (3.7) and (3.8), it has to take into account the functional transformation of continuous variables. Fortunately, unscented transformation [JU96, Julier02] provides good estimates of means and variances for continuous variables subjected to nonlinear transformation. In the proposed algorithm, the unscented transformation plays a key role for computing continuous messages. Specifically, we use it to formulate and compute the π and λ messages since both computations involve conditional probability distribution whose specification may encode nonlinear functions. For completeness, we briefly review the method of unscented transformation next.

3.1.2 Unscented Transformation

Proposed in 1996 by Julier and Uhlmann [JU96], the unscented transformation (UT) is a deterministic sampling method to estimate the mean and variance of a continuous random variable obtained by applying a nonlinear transformation to a random variable with given mean and variance. Consider the following problem: a continuous random variable $\mathbf{x}$ with mean $\bar{\mathbf{x}}$ and covariance matrix $\Sigma_{\mathbf{x}}$ undergoes an arbitrary nonlinear transformation, written as $\mathbf{y} = g(\mathbf{x})$; then what is the mean and covariance of $\mathbf{y}$?

From probability theory, we have

$$p(\mathbf{y}) = \int_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\ dx.$$

However, in general the above integral may be difficult to compute analytically and may not always have a closed-form solution. Therefore, instead of finding the distribution, we retreat to estimating its mean and variance. Based on the principle that it is easier to approximate a probability distribution than an arbitrary nonlinear function, the unscented transformation uses a nonparametric method to estimate the distribution, but leave the original function unchanged. Another method in the literature, called extended Kalman filter (EKF), linearizes the nonlinear function instead. It has been reported that EKF always performs worse than UT in terms of accuracy and computational costs [JU96, JU97, dMer04]. Because of the poor performance of EKF for nonlinear estimation, Julier was joking that linearization methods smell 'bad'. This is why they call the new estimation method for a random variable subject to nonlinear functional transformation Unscented Transformation.

UT uses a minimal set of deterministically chosen sample points called sigma points with the sample mean and the sample covariance equal to the true mean and the true covariance of the prior distribution, respectively. The number of needed sigma points is determined by the dimension of random variable (If it is L-dimensional multivariate distribution, only $2L + 1$ sigma points are needed). Those sigma points are propagated through the original functional transformation individually. Using a Taylor expansion of the function, it can be shown that the posterior mean and covariance calculated using these propagated sigma points are accurate to the 2nd order for any nonlinearity [JU96]. In the special case where the function is linear, the posterior mean and variance are exact.

The original unscented transformation encounters difficulties with high dimensional variables, so the scaled unscented transformation was developed soon afterward [Julier02]. The scaled unscented transformation is a generalization of the original unscented transformation. We will use two terms interchangeably, but both mean scaled unscented transformation in the remainder of this paper.

Now let us describe the formulae of unscented transformation. Assume $\mathbf{X}$ is a L-dimensional multivariate continuous variable with mean vector $\bar{\mathbf{x}}$ and covariance matrix $\Sigma_{\mathbf{x}}$. First, a set of $2L+1$ sigma points denoted as $\mathcal{X}$ are selected by the following scheme:

$$\lambda = \alpha^2(L+\kappa) - L$$

$$\mathcal{X} = \begin{cases} \mathcal{X}_0 & = & \bar{\mathbf{x}} & i = 0 \\ \mathcal{X}_i & = & \bar{\mathbf{x}} + \left(\sqrt{(L+\lambda)\Sigma_{\mathbf{x}}}\right)_i & i = 1, \ldots, L \\ \mathcal{X}_i & = & \bar{\mathbf{x}} - \left(\sqrt{(L+\lambda)\Sigma_{\mathbf{x}}}\right)_i & i = L+1, \ldots, 2L \end{cases} \tag{3.10}$$

and the associated weights for these $2L+1$ sigma points are:

$$\begin{array}{lcll} w_0^{(m)} & = & \frac{\lambda}{L+\lambda} & i = 0 \\ w_0^{(c)} & = & \frac{\lambda}{L+\lambda} + (1 - \alpha^2 + \beta) & i = 0 \\ w_0^{(m)} & = & w_0^{(c)} = \frac{1}{2(L+\lambda)} & i = 1, \ldots, 2L \end{array} \tag{3.11}$$

where α, β, κ are scaling parameters and the superscripts '(m)', '(c)' indicate the weights for computing posterior mean and covariance respectively; i is the index of the sigma point. The values of scaling parameters satisfy $0 \leq \alpha \leq 1$, $\beta \geq 0$ and $\kappa \geq 0$. It has been shown empirically that the specific values chosen for the parameters are not critical because unscented transformation is not sensitive to those parameters. We choose $\alpha = .8$, $\beta = 2$ (optimal for Gaussian prior [Julier02]) and $\kappa = 0$ in all of our experiments.

After selecting sigma points, they are propagated through the functional transformation:

$$\mathcal{Y}_i = \mathbf{g}(\mathcal{X}_i) \quad i = 0, \ldots, 2L \tag{3.12}$$

Finally, the posterior mean and covariance are estimated by combining the propagated sigma points as follows:

$$\bar{\mathbf{y}} \approx \sum_{i=0}^{2L} w_i^{(m)} \mathcal{Y}_i \tag{3.13}$$

$$\Sigma_{\mathbf{y}} \approx \sum_{i=0}^{2L} w_i^{(c)} (\mathcal{Y}_i - \bar{\mathbf{y}})(\mathcal{Y}_i - \bar{\mathbf{y}})^{\mathbf{T}} \tag{3.14}$$

$$\Sigma_{\mathbf{xy}} \approx \sum_{i=0}^{2L} w_i^{(c)} (\mathcal{X}_i - \bar{\mathbf{x}})(\mathcal{Y}_i - \bar{\mathbf{y}})^{\mathbf{T}} \tag{3.15}$$

As a shorthand, we denote the unscented transformation for X undergoing a functional transformation $Y = g(X)$ as the following:

$$(Y.mu, Y.cov) = UT(X \xrightarrow{g(X)} Y) \tag{3.16}$$

In the following, we shall demonstrate the unscented transformation by a simple two-dimensional example. Let $\mathbf{x} = [x_1\ x_2]$ be a Gaussian random variable with mean and covariance matrix given as,

$$\bar{\mathbf{x}} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \qquad \Sigma_{\mathbf{x}} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}.$$

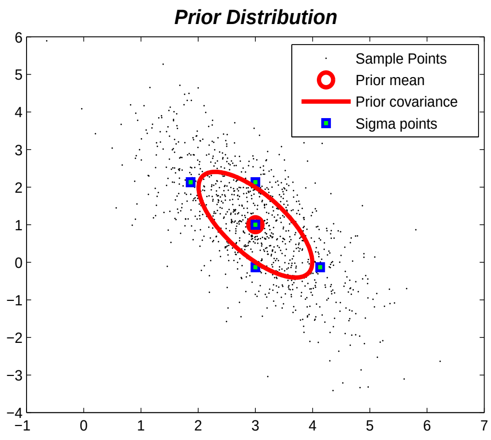
In order to show the robustness of unscented transformation, we choose a set of functions with severe nonlinearity shown as below:

$$y_1 = \log(x_1^2)\cos(x_2) \quad , \quad y_2 = \sqrt{\exp(x_2)}\sin(x_1 x_2)$$

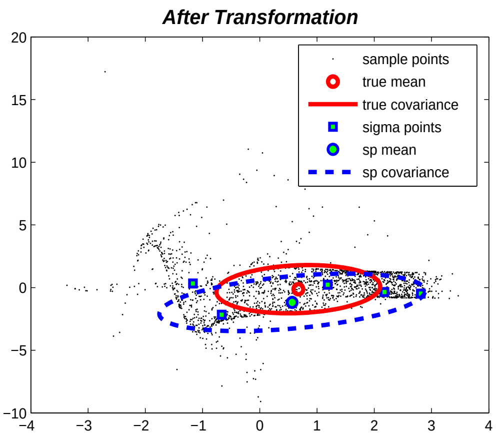
The true posterior statistics are approximated very closely by brute force Monte Carlo simulation using $100,000$ sample points drawn from the prior distribution and then propagated through the nonlinear mapping. We compare them with the estimates calculated by unscented transformation with only 5 sigma points. Figure 3.3 shows that the mean calculated by transformed sigma points is very close to the true mean and the posterior covariance seems consistent and efficient because the sigma-point covariance ellipse is bigger but still tight around the true posterior covariance ellipse.

3.1.3 Message Passing In Continuous Case

As mentioned at the end of Section 3.1.1, message representation and propagation for continuous variables are different from the ones in discrete case. Pearl's conventional equations shown in Equation (3.5) to (3.9), must be modified for continuous variables. First of all, obviously, integral replaces summation in the above equations for continuous variable. Now let us take a closer look at these equations one by one. In recursive Bayesian inference, π message represents prior information and λ message represents



(a) Prior distribution



(b) After nonlinear transformation

Figure 3.3: Demonstration of Unscented Transformation.

evidential support in the form of a likelihood function. Equations (3.5), (3.6), and (3.9) are essentially the combination of different messages by multiplication. They are similar to the data fusion concept where estimates received from multiple sources are combined.

Under the assumption of Gaussian distribution, the fusion formula is relatively straightforward [BLK01]. Specifically, Equations (3.5), (3.6), and (3.9) can be rewritten in terms of the first two moments of the probability distributions as the following:

$$BEL(X)\begin{cases} cov = \left(\frac{1}{\pi(X).cov} + \frac{1}{\lambda(X).cov}\right)^{-1} \\ mu = cov\left[\frac{\pi(X).mu}{\pi(X).cov} + \frac{\lambda(X).mu}{\lambda(X).cov}\right] \end{cases} \tag{3.17}$$

$$\lambda(X)\begin{cases} cov = \left(\sum_{j=1}^{n} \frac{1}{\lambda_{Y_j}(X).cov}\right)^{-1} \\ mu = cov\left[\sum_{j=1}^{n} \frac{\lambda_{Y_j}(X).mu}{\lambda_{Y_j}(X).cov}\right] \end{cases} \tag{3.18}$$

$$\pi_{Y_j}(X) \begin{cases} cov = \left(\dfrac{1}{\pi(X).cov} + \displaystyle\sum_{k \neq j} \dfrac{1}{\lambda_{Y_k}(X).cov} \right)^{-1} \\ mu = cov \left[\frac{\pi(X).mu}{\pi(X).cov} + \displaystyle\sum_{k \neq j} \dfrac{\lambda_{Y_k}(X).mu}{\lambda_{Y_k}(X).cov} \right] \end{cases} \tag{3.19}$$

Equation (3.7) computes the π value for node X. Analytically, this is equivalent to treating X as a functional transformation of $\mathbf{T}$ and the function is the one defined in CPD of X denoted as $h(X)$. We take $\mathbf{T}$ as a multivariate random variable with a mean vector and a covariance matrix; then by using unscented transformation, we obtain an estimate of mean and variance of X to serve as the π value for node X. In Equation (3.7), $\pi_X(T_i)$ is the π messages sent to X from its parent T_i, which is also represented by 'mean' and 'variance'. By combining all the incoming $\pi_X(T_i)$ messages, we can estimate the mean vector and covariance matrix of $\mathbf{T}$. Obviously, the simplest way is to view all parents as independent variables; then take their means to compose the mean vector, and put their variances at the diagonal positions to form a diagonal matrix as the covariance matrix.[1] With that, we can compute the π value of node X by

$$(\pi(X).mu, \pi(X).cov) = UT(\mathbf{T} \xrightarrow{h(X)} X) \tag{3.20}$$

Similarly but a bit more complicated, Equation (3.8) computes the λ message sent to its parent from node X. Note here that we integrate out X and all of its parents except the one (T_i) to which we are sending λ message. Theoretically, this is equivalent to regarding T_i as a functional transformation of X and $\mathbf{T}\backslash T_i$. It is

[1]This is actually how the original loopy algorithm works and why it is not exact. To improve the algorithm, we can estimate the correlations between all parents and pass them into the covariance matrix of $\mathbf{T}$. This could be an interesting future work.

necessary to mention that the function used for transformation is the inverse function of the original one with T_i as the dependent variable. We denote this inverse function as $v(X, \mathbf{T}\backslash T_i)$. Note that this is the key difference from the transformation used in computing $\pi(X)$. In practical applications, there is no guarantee that the inverse function is unique and the function may not be invertible. We assume that functions specified in CPDs in a CHM are invertible. To compute the message, we first augment X with $\mathbf{T}\backslash T_i$ to obtain a new multivariate random variable called $\mathbf{TX}$; then the mean vector and covariance matrix of $\mathbf{TX}$ are estimated by combining $\lambda(X)$ and $\pi_X(T_k)(k \neq i)$. After applying the unscented transformation to $\mathbf{TX}$ with the new inverse function $v(X, \mathbf{T}\backslash T_i)$, we obtain an estimate of the 'mean' and 'variance' for T_i serving as the $\lambda_X(T_i)$ message as below,

$$(\lambda_X(T_i).mu, \lambda_X(T_i).cov) = UT(\mathbf{TX} \overset{v(X,\mathbf{T}\backslash T_i)}{\longrightarrow} T_i) \tag{3.21}$$

With Equations (3.17) to (3.21), we can now compute all messages for continuous variables. For comparison, we summarize the message propagation equations for discrete and continuous cases, respectively, in Table 3.1. These derived propagation equations are one of the key contributions of this dissertation to extend message passing algorithm for continuous variables. As you may notice, unscented transformation plays a key role here. This is why we name this algorithm Unscented Massing Passing for Bayesian Network (UMP-BN).

Message propagation equations for a node X in Bayesian networks.
X has parents $T_1, T_2, \ldots T_m$ and children $Y_1, Y_2, \ldots, Y_n$.

All variables are discrete	All variables are continuous
$BEL(X) = \alpha\pi(X)\lambda(X)$	$BEL(X) \begin{cases} cov = \left(\frac{1}{\pi(X).cov} + \frac{1}{\lambda(X).cov}\right)^{-1} \\ mu = cov\left[\frac{\pi(X).mu}{\pi(X).cov} + \frac{\lambda(X).mu}{\lambda(X).cov}\right] \end{cases}$
$\lambda(X) = \prod_{j=1}^{n} \lambda_{Y_j}(X)$	$\lambda(X) \begin{cases} cov = \left(\sum_{j=1}^{n} \frac{1}{\lambda_{Y_j}(X).cov}\right)^{-1} \\ mu = cov\left[\sum_{j=1}^{n} \frac{\lambda_{Y_j}(X).mu}{\lambda_{Y_j}(X).cov}\right] \end{cases}$
$\pi(X) = \sum_{\mathbf{T}} P(X\mid \mathbf{T}) \prod_{i=1}^{m} \pi_X(T_i)$	$(\pi(X).mu, \pi(X).cov) = UT(\mathbf{T} \xrightarrow{h(X)} X)$
$\lambda_X(T_i) = \sum_{X} \lambda(X) \sum_{T_k:\ k\neq i} P(X\mid\mathbf{T}) \prod_{k\neq i} \pi_X(T_k)$	$(\lambda_X(T_i).mu, \lambda_X(T_i).cov) = UT(\mathbf{TX} \xrightarrow{v(X,\mathbf{T}\setminus T_i)} T_i)$
$\pi_{Y_j}(X) = \alpha\left[\prod_{k\neq j} \lambda_{Y_k}(X)\right]\pi(X)$	$\pi_{Y_j}(X) \begin{cases} cov = \left(\frac{1}{\pi(X).cov} + \sum_{k\neq j} \frac{1}{\lambda_{Y_k}(X).cov}\right)^{-1} \\ mu = cov\left[\frac{\pi(X).mu}{\pi(X).cov} + \sum_{k\neq j} \frac{\lambda_{Y_k}(X).mu}{\lambda_{Y_k}(X).cov}\right] \end{cases}$

Table 3.1: Comparison of message propagation equations.

3.2 Unscented Message Passing Algorithm

The first step of UMP-BN is to initialize the messages. To incorporate the evidence, we allow a λ message sent to a node from itself. From the implementation perspective, we compute 5 different messages for each node in the network: 1. π of the node; 2. λ of the node; 3. λ-from-child message; 4. λ-from-self message; and 5. π-from-parent message. Each message consists of two fields—mean and variance. Recall π message is passed in the top-down direction and λ message is passed from leaf to the root. We update messages for all nodes according to their topological order.

In Pearl's algorithm, the initial message for discrete evidence node is set as a vector of '1' for observed state and 0's for other states. All other messages in discrete network are initialized as a vector of all 1's. For continuous networks, we initialize the message for the continuous evidence node with mean equal to the observed value and variance equal to zero. Similarly, all other messages for continuous variables are initialized as uniform (in particular, zero mean and infinity variance, so-called diffusion prior). After initialization, we then use parallel updating for all nodes based on the equations (3.18) to (3.21). Message passing continues till the posterior belief has converged or a pre-defined maximum number of iterations is reached. We assess the convergence by checking if the belief change is less than a pre-specified threshold (say, 10^{-4}).

The general UMP-BN algorithm is summarized in Figure 3.4.

3.3 Numerical Experiment

To test the algorithm, we use two popular real-world networks each has multiple loops with different sizes. One is INCINERATOR, borrowed from [Lau92] in which the author proposed the Junction Tree algorithm for conditional linear Gaussian network (CLG). Another one is ALARM, used in the paper [BSC89] to compare various inference algorithms. These networks are either CLGs or discrete networks originally.

```
Algorithm: Unscented Message Passing for arbitrary
           continuous Bayesian networks (UMP-BN).
Input: Arbitrary continuous Bayesian network given
       a set of evidence
Output: Estimated mean and variance of posterior
        distribution for every hidden continuous node.

1. Order the nodes in the network according to their topological order.

2. Specify the iteration number/convergence condition.

3. Initialize messages for all node

4.   while Not Converge do
         Run parallel message updating for all hidden nodes using Equations (3.18) to
         (3.21).
     end while

5. Compute the posterior beliefs in terms of estimated mean and variance for every
   hidden node using Equation (3.17).
```

Figure 3.4: Unscented Message Passing Algorithm

In our experiments, we leave the network structures unchanged but specify different types of CPDs to see how the algorithm performs under different circumstances. The network structures are shown in Figure 3.5, and 3.6 respectively. In our experiments, we assume all leaf nodes in the network are observable evidence.

We use normalized error and Kullback-Leibler (KL) divergence as the performance measures in all of the experiments. Normalized error is defined as the ratio of the absolute error over the reference true value. It will be used in evaluating the accuracy

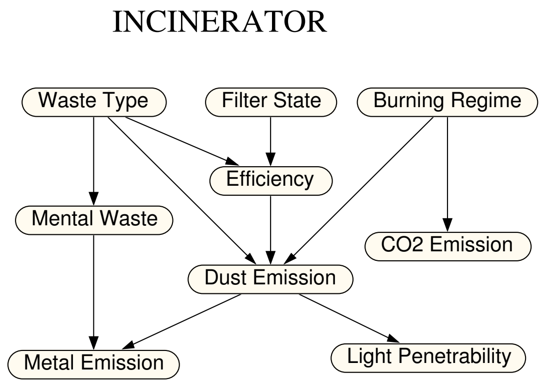


Figure 3.5: INCINERATOR

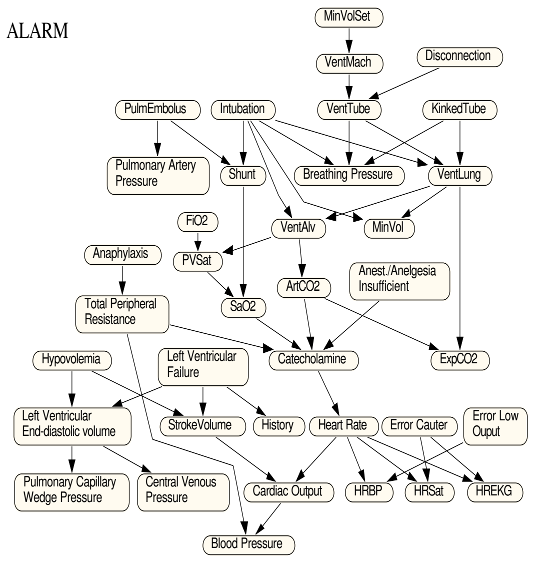


Figure 3.6: ALARM: a network for monitoring patients in intensive care.

of estimation for mean and variance of the posterior distribution. KL divergence is the well-known measure to assess the similarity between two distributions defined as:

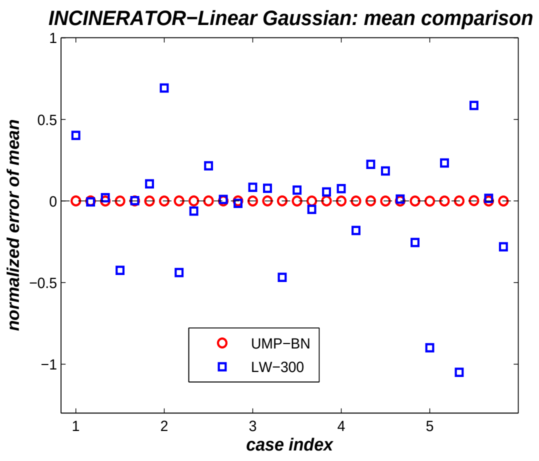
$$\mathcal{D}_{KL}(P||Q) = \int_x p(x)log\frac{p(x)}{q(x)}dx$$

where $p(x), q(x)$ are two different probability density functions.

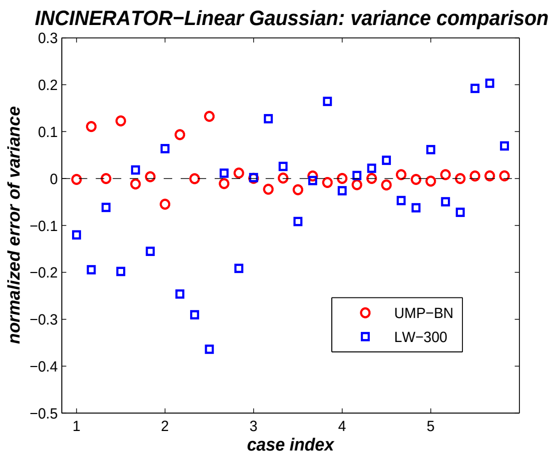
3.3.1 Case 1: Linear Gaussian

In this experiment, we first randomly generate linear Gaussian CPDs for all nodes in the network so that the exact inference algorithm, Junction tree, can be used to provide a true answer as the gold standard. We then sample the network and clamp the evidence nodes with the sampled values. Given a set of evidence, we run UMP-BN to compute the mean and variance of the posterior distribution for every hidden continuous node. For comparison purpose, we also implement a sampling algorithm, likelihood weighting (LW) [FC89, SP90] with various sample size.

For INCINERATOR, We found UMP-BN always converges with an average number of iterations about 9.2. Figure 3.7 shows the performance comparison in means and variances of the posterior distributions for all hidden variables in 5 random runs. We use 300 samples in LW so that the total amount of computation time is roughly the same. As can be seen from the figure, UMP-BN always provides accurate estimates of means, while the estimated variances deviate from the true somewhat but still are better than LW in most cases. We also compare the average KL divergence between the estimated distributions and the true ones. The results are summarized in Table 3.2. It shows that UMP-BN performs almost two orders of magnitude better than LW with roughly the same computation time.



(a) mean comparison



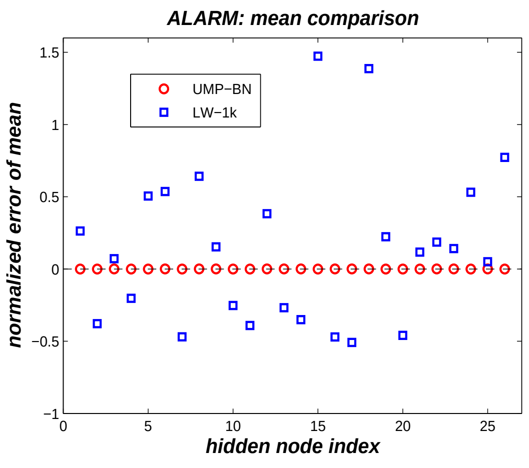
(b) variance comparison

Figure 3.7: INCINERATOR-Linear Gaussian: Performance comparison. The ground true is provided by Junction tree.

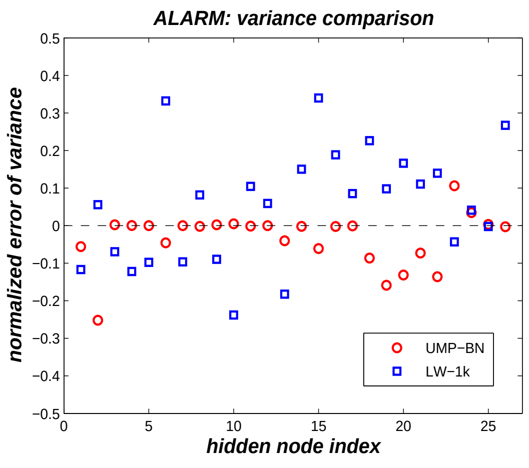
Table 3.2: INCINERATOR: Average KL-divergence.

	Average KL Divergence
UMP-BN	0.0017
LW-300	0.0711

For ALARM model, there are 11 leaf evidence nodes and 26 hidden nodes. Using 300 samples, LW performed very poorly in this case because the sample space is much bigger than the one for the model INCINERATOR. Therefore, we use $1,000$ samples instead for LW. With this sample size, LW takes about three times the computation time of UMP-BN. We found UMP-BN always converges in this case with an average number of iterations about 15.8. Figure 3.8 illustrates that UMP-BN still performs significantly better than LW with $1,000$ samples in a typical run. We implement 5 random trials and the summarized average KL divergence is shown in Table 3.3.



(a) mean comparison



(b) variance comparison

Figure 3.8: ALARM-Linear Gaussian: Performance comparison. The ground true is provided by Junction tree.

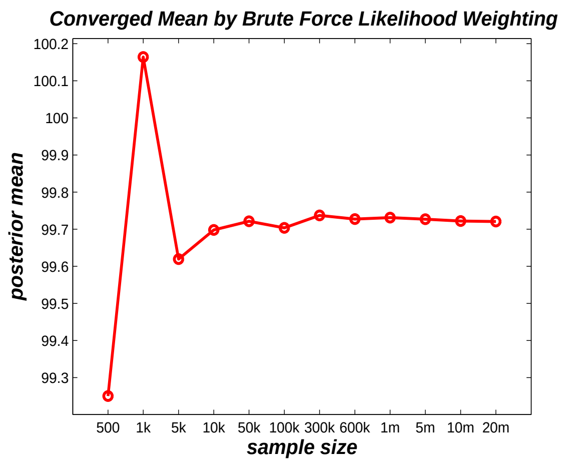
Table 3.3: ALARM: Average KL-divergence.

	Average KL Divergence
UMP-BN	0.0084
LW-1k	0.0576

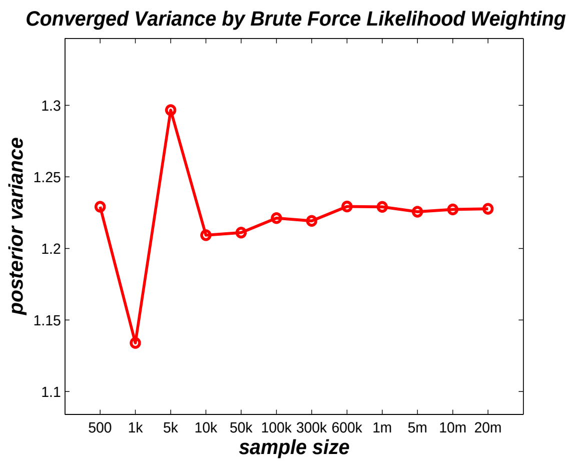
3.3.2 Case 2: Nonlinear Gaussian

Note that UMP-BN is targeted for probabilistic inference dealing with arbitrary continuous Bayesian network. To test the algorithm robustness in the nonlinear case, we purposely specify severe nonlinear functional relationships for continuous variables listed as below:

$$\begin{aligned}
\mathbf{F}(\mathit{FilterState}) &\sim \mathcal{N}(-10,\ 3)\\
\mathbf{W}(\mathit{WasteType}) &\sim \mathcal{N}(100,\ 10)\\
\mathbf{B}(\mathit{BurningRegime}) &\sim \mathcal{N}(50,\ 5)\\
\mathbf{E}(\mathit{Efficiency}) &\sim \mathcal{N}(W+2F,\ 1)\\
\mathbf{C}(\mathit{CO_2Emission}) &\sim \mathcal{N}(e^{\sqrt[3]{B}},\ 3)\\
\mathbf{D}(\mathit{DustEmission}) &\sim \mathcal{N}(\sqrt{W}\times\log(E)-B, 5)\\
\mathbf{Min}(\mathit{MentalWaste}) &\sim \mathcal{N}(\sqrt{W}+6,\ 3)\\
\mathbf{Mout}(\mathit{MentalEmission}) &\sim \mathcal{N}(0.5\times D\times Min,\ 5)\\
\mathbf{L}(\mathit{LightPenetrability}) &\sim \mathcal{N}(-5\times D,\ 5)
\end{aligned}$$



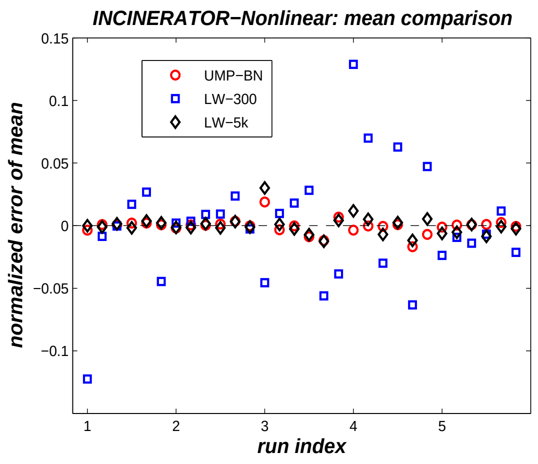
(a) mean convergence



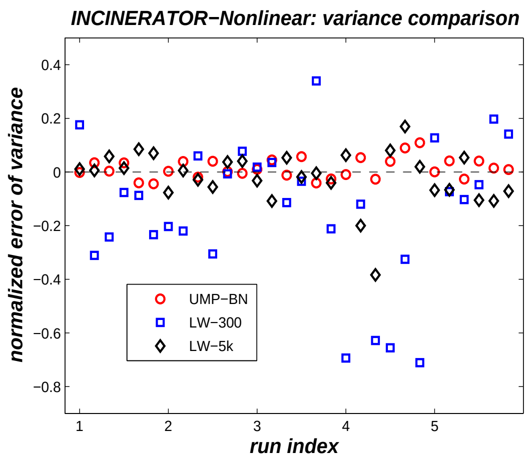
(b) variance convergence

Figure 3.9: Convergence demonstration by brute force Likelihood Weighting.

Since no exact inference algorithm is available in this case, we use LW with a very large sample size to compute an approximate true solution as the reference base. To make sure the inference results are converged, we implement LW with monotonically increasing sample sizes. Figure 3.9 shows that LW with a few millions samples will start to converge for both mean and variance of the posterior distribution. To be conservative, we use LW with 20-million samples to compute the approximate true answers in our experiment.



(a) mean comparison



(b) variance comparison

Figure 3.10: INCINERATOR-Nonlinear: Performance Comparison. The ground true is well approximated by LW with 20-million samples.

Given each of the five sets of random observations generated by sampling the network, we run UMP-BN and LW with 300 and $5,000$ samples respectively. The experiment results show that UMP-BN performs significantly better than LW due to its robust estimation under nonlinear transformations. From Figure 3.10, it is obvious that even with $5,000$ samples (an order of magnitude more computation time than UMP-BN), LW still performs worse. Furthermore, the average KL divergence summarized in Table 3.4 shows the clear superiority in performance of UMP-BN. On the other hand, LW has the advantage of being a model-independent algorithm. But when the model space is large or when the evidence is unlikely, LW needs a large number of samples to converge which could be a significant issue for time-critical applications.

Table 3.4: Nonlinear INCINERATOR: KL comparison.

	Average KL Divergence
UMP-BN	0.0056
LW-300	1.4965
LW-5k	0.0264

3.4 Summary

In this chapter, we propose a novel algorithm to solve the difficult inference problem when nonlinear CPDs and/or non-Gaussian distributions are involved for the continuous variables in the Bayesian networks. The new algorithm, UMP-BN, uses the first two moments of the probability distribution to represent the continuous message and unifies the message passing framework with effective nonlinear transformation method. In addition, the method is not limited to Gaussian distributions. For general continuous distribution, it can always produce the estimates of the first two moments. The algorithm is very computationally efficient because we use deterministic sampling method to capture the features of the distribution so that it is scaled linearly to the size of the network.

For general hybrid Bayesian networks where both discrete and continuous variables are present, we proposed a general message passing framework [SC07b] discussed in the next chapter (Chapter 4). In the framework, a set of the interface nodes will be introduced to partition the network into independent segments so that different appropriate algorithms (exact or approximate) can be applied for each network segment. For difficult network segments such as those involving nonlinear and/or non-Gaussian variables, UMP-BN algorithm proposed in this chapter can be applied in a straightforward manner.

One interesting future work based on UMP-BN is to integrate the algorithm with the stochastic sampling methods. While UMP-BN provides good estimates for mean and variance, the true underlying distribution may have multiple modes. And practically, it might be more important to know where the probability mass is than just knowing the first two moments. To do so, one idea is to take the intermediate messages computed in UMP-BN to develop a good importance function for sampling. This could dramatically improve the sampling efficiency.

Chapter 4: Message Passing for General Hybrid Bayesian Networks

4.1 Introduction

We described in detail how to implement message passing for arbitrary continuous BN in Chapter 3. In this chapter, we will discuss an extension of message passing for hybrid BN models. In a hybrid BN, because of the differences in message representation and manipulation for discrete and continuous variable, there is no simple and efficient way to pass messages between them. For example, in [YD06], the authors use general nonparametric form to represent messages and formulate their calculation by numerical integrations for hybrid models. Their method includes many functional estimations, sampling, numerical integrations and so it is very computationally intensive.

Essentially, messages are likelihoods or probabilities. In the discrete case, messages are represented and manipulated by probability vectors and conditional probability tables (CPTs), which is relatively straightforward. For continuous variables, as described in Chapter 3, we use the first two moments, the mean and variance of a probability distribution, to represent the messages of a continuous variable regardless of its distribution. This simplification makes message calculation and propagation efficient between continuous variables while keeping key information about the original distributions. Furthermore, to deal with possibly arbitrary functional relationships between continuous variables, we propose to use unscented transformation [JU96, Julier02] to derive the corresponding messages. Unscented transformation is very efficient in estimating the first two moments for the transformed continuous variable through nonlinear function. For arbitrary continuous network, the approach we called unscented message passing (UMP) in Chapter 3 works very well [SC07a]. But in hybrid model, message propagation between discrete and continuous variables is very difficult to do directly due to their different formats. To deal with this issue, we propose to apply conditioning. First we partition the original hybrid Bayesian network into separate discrete and continuous network segments by conditioning on discrete parents of continuous variables [SC07b]. We can then process message passing separately for each network segment before final integration.

One of the benefits to partitioning the network is to ensure that there is at least one efficient inference method applicable to each network segment. In conditional hybrid model (CHM) defined in Chapter 2, a continuous node is not allowed to have any discrete children. Therefore, the original networks can be partitioned into separate parts by the discrete parents of continuous variables. We call these nodes the interface nodes. Each network segment separated by the interface nodes consists of pure discrete or continuous variables. By conditioning on interface nodes, the variables in different network segments are independent of each other. We then can perform inference using possible different algorithms for separate sub-networks. Finally, intermediate results computed in different segments are integrated through the interface nodes. We then estimate the posterior distribution of every hidden variable given evidence in all network segments.

In this chapter, we present a hybrid message passing algorithm for CHMs. Next, we will describe the methods of network partition and hybrid message passing by conditioning and integration. Numerical simulations results show promising performance of the proposed hybrid message passing algorithm. At the end of this chapter, we will discuss the algorithm complexity and potential research directions to improve the algorithm.

4.2 Hybrid Loopy Propagation Algorithm

4.2.1 Network Partition

First of all, as mentioned earlier, we know that in CHM, any discrete node can only have discrete parent nodes in the hybrid models, which implies continuous variable can not have any discrete child node. Two definitions are given as below:

Definition 4.1 (Discrete parent)**.** *In a conditional hybrid Bayesian network model (CHM), a discrete variable is called a discrete parent if and only if it has at least one continuous child node.*

Definition 4.2 (Interface node)**.** *The set of all discrete parent nodes in a CHM is called the interface nodes of the network.*

It is well-known that Bayesian networks have an important property that every node is independent of its non-descendant nodes given its parents. Therefore the following theorem follows:

Theorem 4.1. *Let $\mathcal{I}$ be the set of interface nodes in a CHM. Then $\mathcal{I}$ d-separates all continuous nodes from other non-interface discrete nodes.*

Proof:

Let us first review the definition of d-separation [Pearl88, p.117]:

Definition 4.3 (d-separation)**.** *if* $\mathbf{X}, \mathbf{Y}$*, and* $\mathbf{Z}$ *are three disjoint subsets of nodes in a DAG* $\mathcal{D}$*, then* $\mathbf{Z}$ *is said to* **d-separate X** *from* $\mathbf{Y}$*, if there is no path between a node in* $\mathbf{X}$ *and a node in* $\mathbf{Y}$ *along which the following two conditions hold: (1) every node with converging arrows is in* $\mathbf{Z}$ *or has a descendent in* $\mathbf{Z}$ *and (2) every other node is outside* $\mathbf{Z}$*.*

If a path satisfies the condition above, it is said to be active; otherwise, it is said to be blocked by $\mathbf{Z}$.

Let us denote the set of all non-descendant nodes of $\mathcal{I}$ as $\mathcal{S}_k$. By definition, in a CHM, there is no continuous parent for any discrete node. It follows that $\mathcal{S}_k$ must be pure discrete because if there were continuous nodes in $\mathcal{S}_k$, the continuous nodes should be descendant nodes of $\mathcal{I}$.

Because a CHM is a hybrid BN model, there must be continuous descendant nodes of $\mathcal{I}$. And the $\mathcal{I}$'s continuous descendant nodes may have continuous predecessor nodes that are not $\mathcal{I}$'s descendant. We denote the set of all non-predecessor continuous nodes of $\mathcal{I}$ as $\mathcal{S}_i$. Similarly, $\mathcal{I}$ may have discrete descendant nodes, and the $\mathcal{I}$'s discrete descendant nodes may have discrete predecessor nodes that are not the descendant of $\mathcal{I}$. We denotes the set of non-predecessor discrete nodes as $\mathcal{S}_j$. By definition of the CHM, $\mathcal{S}_i$ can not have any descendant node in $\mathcal{S}_j$ and vice versa. This is because if there were some nodes in $\mathcal{S}_j$ that are parents of nodes in $\mathcal{S}_i$, then these discrete parent nodes should be in $\mathcal{I}$. Therefore, $\mathcal{I}$, $\mathcal{S}_i$, and $\mathcal{S}_j$ include all nodes in a CHM and they are disjoint sets. Figure 4.1 shows the relationships of these sets clearly.

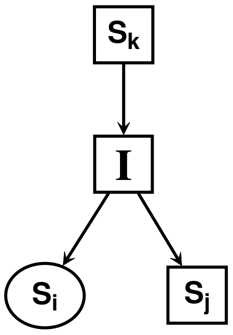


Figure 4.1: A partitioned conditional hybrid Bayesian network.

According to Definition 4.3, any path between variable in $\mathcal{S}_i$ and variable in $\mathcal{S}_k$ is blocked by $\mathcal{I}$. Then, $\mathcal{S}_i$ is d-separated from $\mathcal{S}_k$ by $\mathcal{I}$. Similarly, $\mathcal{S}_j$ is d-separated from $\mathcal{S}_k$ by $\mathcal{I}$; and $\mathcal{S}_i$ is d-separated from $\mathcal{S}_j$ by $\mathcal{I}$. Therefore, $\mathcal{S}_i$, $\mathcal{S}_j$, and $\mathcal{S}_k$ are pairwisely d-separated by $\mathcal{I}$. This completes the proof.

The main purpose of introducing interface nodes is to separate the network segments consisting of pure continuous variables from a original CHM. Then given the interface nodes, any continuous variable is conditionally independent with other discrete variables. By this partition, we can choose the most appropriate inference algorithm such as UMP-BN for the separated continuous sub-networks. For discrete network segments, it is better to combine them together for computational convenience because many well established inference algorithms work for pure discrete BNs. Therefore, although interface nodes partition the discrete variables as well, we consider the discrete sub-networks including the interface nodes as a network segment. An example is shown in Figure 4.2 where a 13-node hybrid model is presented. Following the convention, we use squares or rectangles to depict discrete variables and circles or ellipses to depict continuous variables. As can be seen, K, A and C

are the interface nodes in this example. The arcs between discrete parents and their continuous children are shown as dotted lines. It can be seen that the interface nodes partition all continuous variables from the network. We have four separate network segments in this example — two discrete parts $\{H, B, F, K, G\}$ and $\{J, A, C\}$ and two continuous parts $\{T, R, S\}$ and $\{X, Y\}$.

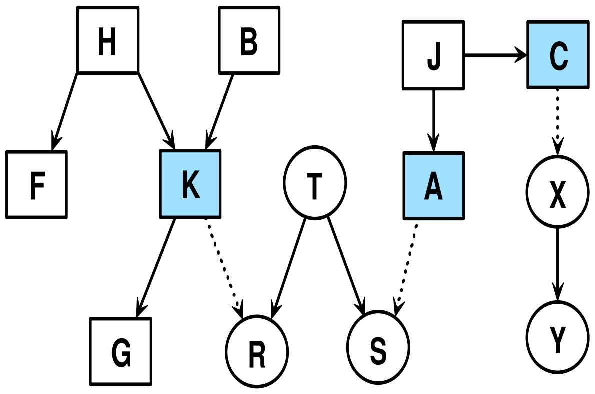


Figure 4.2: Demonstration of interface nodes and network partition

After the network is partitioned by the interface nodes, we choose the most appropriate inference algorithm for each network segment. In fact, we can also combine some network segments if the same algorithm works for all of them. Typically, continuous network segments with nonlinear and/or non-Gaussian CPDs are the most difficult ones to deal with. For these segments, we can apply the unscented message passing algorithm (UMP-BN) [SC07a], described in the previous chapter, to provide an alternative for approximate solutions.

Finally, we need to summarize the prior and evidential information for each network segment and encode it as messages to be propagated between network segments through the interface nodes. This is similar to Pearl's message passing algorithm and can be integrated with the UMP-BN loopy algorithm mentioned above in a unified manner.

4.2.2 Hybrid Message Passing Algorithm

For a hybrid model, without loss of generality, let us assume that the network is partitioned into two parts denoted as $\mathcal{D}$ and $\mathcal{C}$. Part $\mathcal{D}$ is a discrete network including the interface nodes and it is solvable by appropriate algorithms such as junction tree or discrete loopy propagation. Part $\mathcal{C}$ is an arbitrary continuous network. Let us denote the observable evidence in part $\mathcal{D}$ as $\mathbf{E_d}$, and the evidence from $\mathcal{C}$ as $\mathbf{E_c}$. Therefore the entire evidence set $\mathbf{E}$ consists of $\mathbf{E_d}$ and $\mathbf{E_c}$. As mentioned before, given interface nodes, variables from the two network segments are independent of each other. The evidence from part $\mathcal{D}$ affects the posterior probability of hidden nodes in part $\mathcal{C}$ only through the channel of the interface nodes and vice versa.

We therefore summarize the prior and evidence information of each network segment and encode it as either π or λ of the interface nodes. Assuming that the set of interface nodes between two network segments is $\mathbf{I}$, then $\lambda(\mathbf{I}) = P(\mathbf{E_c}|\mathbf{I})$ and $\pi(\mathbf{I}) = P(\mathbf{I}|\mathbf{E_d})$. These messages are to be passed between network segments to facilitate information integration. As in Pearl's algorithm, this approach can be easily integrated with the UMP-BN loopy algorithm mentioned above in a unified manner.

We use the following concrete example to illustrate how to integrate messages from different network segments. As can be seen in Figure 4.3, synthetic hybrid model-1 has K as the interface node which divides the network into a discrete part consisting of H, B, F, K, G and a continuous part consisting of T, R, S, M, Y. For the purpose of illustration, let us assume all discrete nodes are binary and all continuous nodes are scalar Gaussian variables.

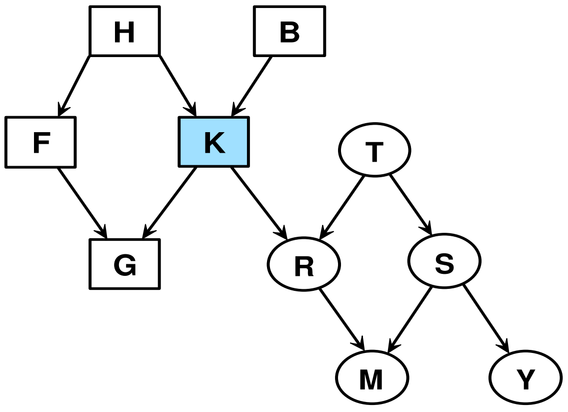


Figure 4.3: Synthetic hybrid Bayesian networks-1.

Suppose the leaf nodes G, M, Y are observable evidence. We will first focus on the continuous network segment. In this step, we will compute λ message sent to the interface node K to propagate evidential information from continuous nodes. And conditioning on each possible state of K, we estimate the posterior distributions for all hidden continuous variables given continuous evidence. Under Gaussian assumption, these posterior distributions are represented by means and variances and they are intermediate results that will be combined after we obtain the *a posterior* probability distribution of the interface node K given all evidence. Probabilities of all possible states of K serve as the mixing weights, similar to computing the mean and variance of a Gaussian mixture.

Given K, it is relatively straightforward to compute the likelihood of continuous evidence $M = m, Y = y$ because we can estimate the conditional probability distribution of evidence node given interface nodes and other observations by applying an appropriate inference algorithm for continuous Bayesian networks such as UMP-BN.

For example, let

$$L(M = m, Y = y|K = 1) = a$$

$$L(M = m, Y = y|K = 2) = b \ ,$$

then to incorporate the evidence likelihood is equivalent to adding a binary discrete dummy node as the child of the interface node K with the conditional probability table shown as the following:

	Dummy	
K	1	2
1	αa	1-αa
2	αb	1-αb

where α is a normalization constant.

By setting 'Dummy' to be observed as state 1, the continuous segment could be replaced entirely by one discrete node 'Dummy'. Dummy encodes the influence of the continuous evidence to the discrete network segment. Then the original hybrid Bayesian network can be transformed into a pure discrete model shown in Figure 4.4 in which 'Dummy' integrates all of continuous evidence information.

The second step is to compute the posterior distributions for all hidden discrete nodes given $G = g, Dummy = 1$. We have several algorithms to choose for inference depending on the complexity of the transformed model. In general, we can always apply discrete loopy algorithm to produce approximate results regardless of network topology. Note that the posterior distributions of the discrete nodes have taken into account all evidence including the ones from continuous segment via the 'Dummy' node. Next, we need to send the updated information back to the continuous sub-network via the set of interface nodes. This is done by computing the joint posterior

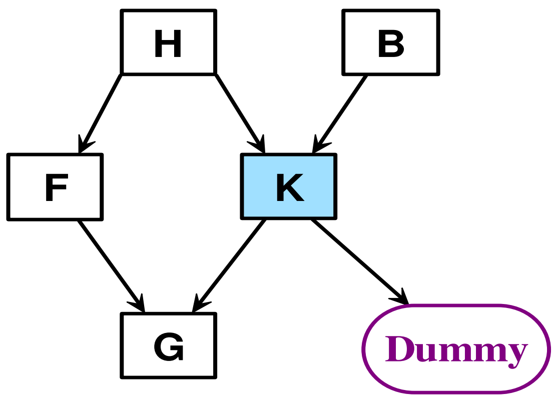


Figure 4.4: Transformed model with dummy node.

probability distribution of the interface nodes denoted as $P(\mathbf{I}|\mathbf{E})$. It is essentially the π messages to be sent to the continuous network segment.

With the messages encoded in the interface nodes, the last step is to go back to the continuous segment to compute the *a posterior* probability distributions for all hidden continuous variables. Recall that in the first step, for any hidden continuous variable X, we already have $P(X|\mathbf{I}, E_c)$ computed and saved. The following derivation shows how to compute $P(X|\mathbf{E})$:

$$\begin{aligned} P(X|\mathbf{E}) &= P(X|E_c, E_d) \\ &= \sum_{\mathbf{I}} P(X, \mathbf{I}|E_c, E_d) \\ &= \sum_{\mathbf{I}} P(X|\mathbf{I}, E_c, E_d) P(\mathbf{I}|E_c, E_d) \\ &= \sum_{\mathbf{I}} P(X|\mathbf{I}, E_c) P(\mathbf{I}|\mathbf{E}) \end{aligned} \tag{4.1}$$

The fourth equality is due to the fact that the set of interface node d-separates the node X from E_d.

Suppose that given an instantiation of the set of interface nodes $\mathbf{I}=i$, the conditional probability distribution $P(X|\mathbf{I}=i, E_c)$ is a Gaussian distribution with mean $\bar{x}_i$ and variance σ_i^2. Then Equation (4.1) is equivalent to computing the probability density function of a Gaussian mixture with $P(\mathbf{I}=i|\mathbf{E})$ as the weighting factors. Denoting $P(\mathbf{I}=i|\mathbf{E})$ as p_i, the mean $\bar{x}$ and the variance σ_x^2 of $P(X|\mathbf{E})$ can be computed as the following [BLK01, p56]:

$$\bar{x} = \sum_i p_i \bar{x}_i \tag{4.2}$$

$$\sigma_x^2 = \sum_i p_i \sigma_i^2 + \sum_i p_i \bar{x}_i^2 - \bar{x}^2 \tag{4.3}$$

In summary, the overall algorithm of hybrid message passing for general mixed Bayesian networks (HMP-BN) is presented in Figure 4.5.

4.3 Numerical Experiments

4.3.1 Experiment Method

We use two synthetic hybrid models for experiments. One is the model shown in Figure 4.3. We call it GHM-1 and used it as an example to illustrate the algorithm in Section 4.2.2. GHM-1 has one loop in each network segment respectively (partitioned by the interface node K). Another experimental model, called GHM-2, is shown in Figure 4.6. GHM-2 has multiple loops in the continuous segment.

For GHM-1, we assume that the leaf nodes G, M, Y are observable evidence. We model its continuous segment as a linear Gaussian network given the interface node

Algorithm: Hybrid Message Passing for General Mixed
Bayesian Network (HMP-BN).
Input: General hybrid Bayesian network given a set of evidence
Output: Posterior marginal distributions of all hidden nodes.

1. Determine the interface nodes and partition the network into independent segments by interface nodes. Choose the appropriate inference algorithm for each network segment.

2. For each continuous network segment: compute the λ message to send to the interface nodes and the intermediate posterior distribution of the hidden continuous variables given the interface nodes and the local evidence.

3. Transform the original network into an equivalent discrete model. In this new model, each continuous network segment is replaced by a dummy node. And every dummy node is added as a child of all discrete parents of the corresponding continuous network segment. Dummy discrete nodes carry the λ message from continuous evidence to the interface nodes.

4. Compute the posterior distribution for every hidden discrete variable using the transformed discrete model. The joint posterior probability table of the discrete parents of each continuous network segment is saved as the π message to be sent back to the associated continuous network segment.

5. Compute the posterior distribution for every hidden continuous variable given all evidence by integrating the final π messages of the interface nodes.

Figure 4.5: Hybrid Message Passing Algorithm for General Mixed Bayesian Network

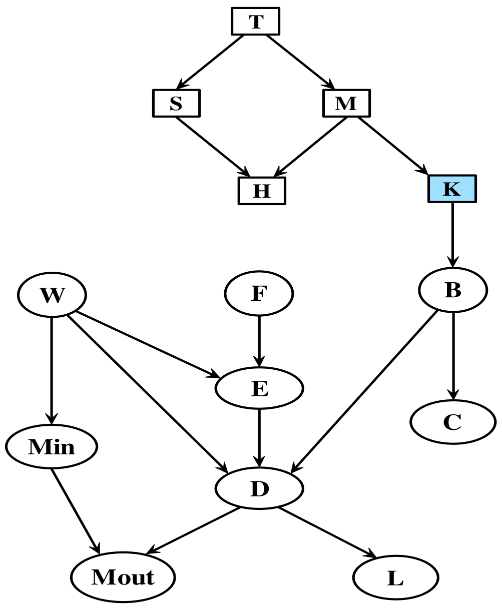


Figure 4.6: GHM-2

K. Therefore the original network is a CLG so that an exact inference algorithm (such as Junction Tree) can be used to provide the true answer for performance evaluation. The CPTs and CPDs for nodes in GHM-1 are randomly specified.

Note that our algorithm can handle general arbitrary hybrid models, not just CLG. GHM-2 is designed specifically to test the algorithm in the case where nonlinear CPDs are involved in the model. The structure of the continuous segment in GHM-2 is borrowed from [Lau92] in which the author proposed the Junction Tree algorithm for CLG. The discrete nodes in the GHM-2 are binary and we randomly specify the CPTs for them similarly as in GHM-1. But the CPDs for the continuous nodes are deliberately specified using severe nonlinear functions to test the robustness of the algorithm. These distributions are:

$$\begin{aligned}
\mathbf{F} &\sim \mathcal{N}(-10,\ 3)\\
\mathbf{W} &\sim \mathcal{N}(100,\ 10)\\
\mathbf{B|K=1} &\sim \mathcal{N}(50,\ 5)\\
\mathbf{B|K=2} &\sim \mathcal{N}(60,\ 5)\\
\mathbf{E} &\sim \mathcal{N}(W+2F,\ 1)\\
\mathbf{C} &\sim \mathcal{N}(e^{\sqrt[3]{B}},\ 3)\\
\mathbf{D} &\sim \mathcal{N}(\sqrt{W}\times\log(E)-B, 5)\\
\mathbf{Min} &\sim \mathcal{N}(\sqrt{W}+6,\ 3)\\
\mathbf{Mout} &\sim \mathcal{N}(0.5\times D\times Min,\ 5)\\
\mathbf{L} &\sim \mathcal{N}(-5\times D,\ 5)
\end{aligned}$$

We assume that the evidence set in the GHM-2 is $\{H, C, Mout, L\}$. Since no exact algorithm is available for such model, we use brute force sampling method, likelihood weighting, to obtain an approximate true solution with 20-million samples.

In our experiments, we first randomly sample the network and clamp the evidence nodes by their sampled value. Then we run the HMP-BN to compute the posterior distributions for the hidden nodes. It is important to mention that in both discrete and continuous network segments, we implement HMP-BN using loopy algorithms to make it general, although Junction tree could be used in network segment whenever it is applicable. In addition, we run LW using as many samples as it can generate within the roughly same amount of time HMP-BN consumes. There are 10 random runs for GHM-1 and 5 random runs for GHM-2. We compare the average Kullback-Leibler (KL) divergence of the posterior distributions by different algorithms.

Given unlikely evidence, it is well-known that the sampling methods converge very slowly even with large sample size. We use GHM-1 to test the robustness of our algorithm in this case because Junction tree can provide ground truth for GHM-1 no matter how unlikely its evidence is. We generate 10 random cases with evidence likelihood between $10^{-5} \sim 10^{-15}$ and run both HMP-BN and LW to compare the performances.

4.3.2 Experiment Results

For model GHM-1, there are 4 hidden discrete nodes and 3 hidden continuous nodes. Figure 4.7 illustrates the posterior probabilities of hidden discrete nodes computed by Junction tree, HMP-BN and LW in two typical runs. Since GHM-1 is a simple model and we did not use unlikely evidence, both HMP-BN and LW perform well.

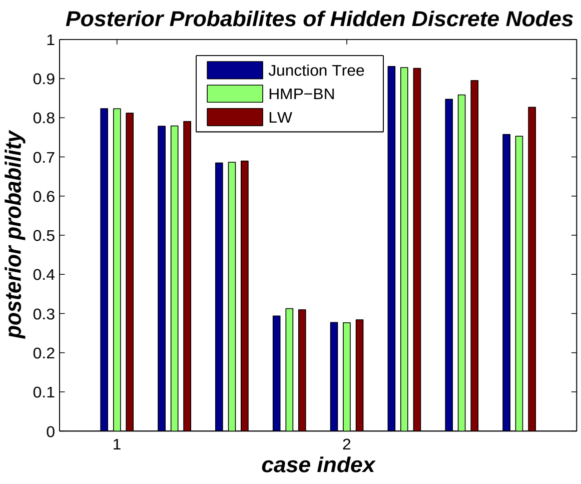
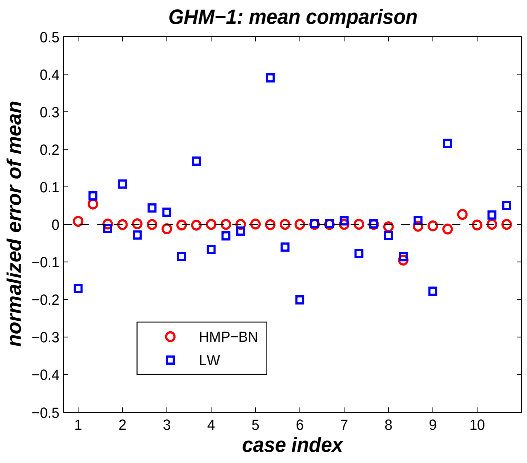


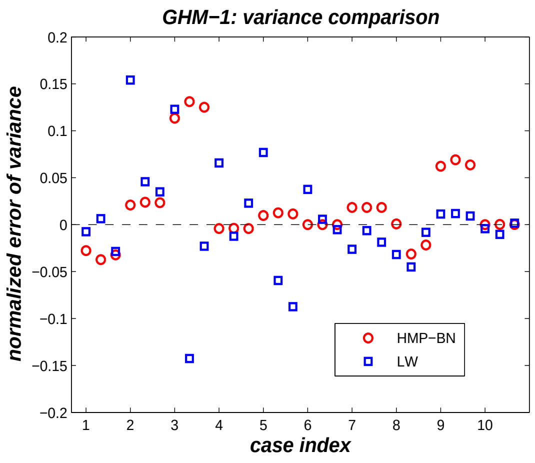
Figure 4.7: Posterior probability of hidden discrete variables in two typical runs.

For continuous variables in GHM-1, Figure 4.8 shows the performance comparisons in means and variances of the posterior distributions for the hidden continuous nodes in all of the 10 runs. The normalized error is defined as the ratio of the absolute error over the reference true value. From the plot, it is evident that HMP-BN provides accurate estimates of means, while the estimated variances deviate from the true somewhat but HMP-BN is still better than LW in most cases.

We then demonstrate the robustness of HMP-BN by testing its performance given



(a) mean comparison



(b) variance comparison

Figure 4.8: GHM-1 Performance Comparison for 10 random runs. (the ground true is provided by Junction tree).

unlikely evidence shown in Figure 4.9. In this experiment, 10 random sets of evidence are chosen with likelihood between 10^{-5} and 10^{-15}. As can be seen, HMP-BN performs significantly better than LW in this case. The average KL divergence are consistently small with the maximum value less than 0.05. This is not surprising because LW uses the prior to generate samples so that it hardly hits the area close to evidence.

We summarize the performance results with GHM-1 in Table 4.1. Note that given unlikely evidence, the average KL divergence by HMP-BN is more than one order of magnitude better than LW.

In GHM-2, due to the nonlinear nature of the model, no exact method exists to provide a benchmark. We use LW with 20-million samples to obtain an approximation to the true value. We implemented five simulation runs with randomly sampled evidence. In this experiment, we employed the UMP-BN algorithm presented in Chapter

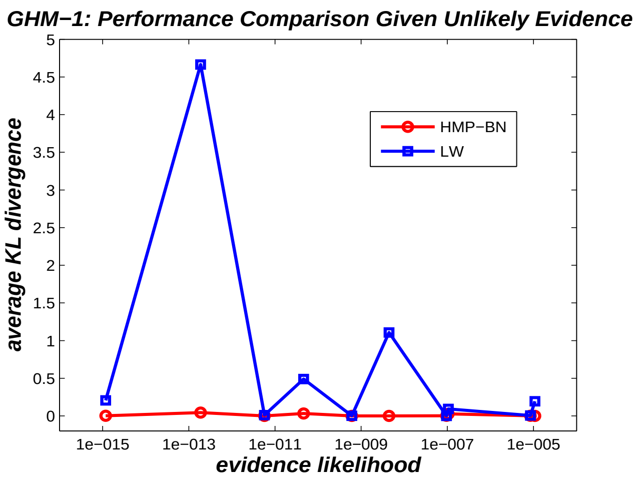


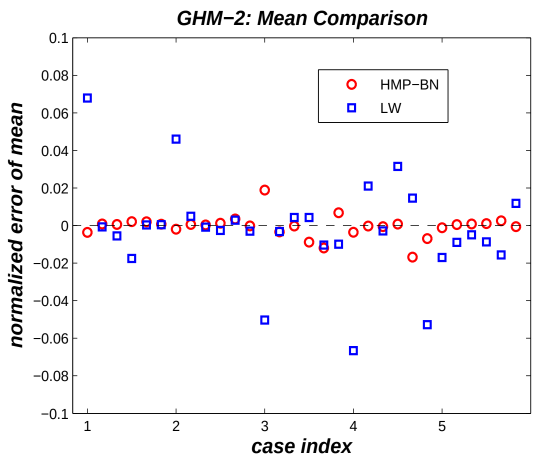
Figure 4.9: GHM-1: Performance Comparison Given Unlikely Evidence.

Average KL divergence	Normal Evidence $> 10^{-5}$	Unlikely Evidence $10^{-5} \sim 10^{-15}$
HMP-BN	0.0011	0.0108
LW	0.0052	0.67

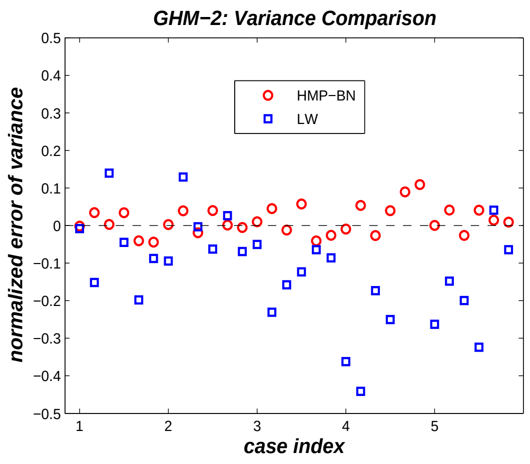
Table 4.1: Average KL-divergence Comparison in Testing GHM-1

3 for inference in the continuous network segment. To evaluate the performance of this new algorithm by hybrid messages passing, we implement LW with as many samples as it can generate in the roughly same time HMP-BN takes. Figure 4.10 shows the performance comparison in means and variances of the posterior distribution for the hidden continuous variables. Note that the benchmark is provided by LW with 20-million samples. Also, Table 4.2 summarizes the average KL divergence in testing GHM-2. From the data, we see that HMP-BN combining with UMP-BN applied in the continuous sub-network produces very good results. In this nonlinear model with typical evidence, the new algorithm performs much better than LW despite its

advantages of being a model-free algorithm. However, we have to admit there is some advantage for HMP-BN in these examples because there is only one interface node in these models.



(a) mean comparison



(b) variance comparison

Figure 4.10: GHM-2 Performance Comparison for 5 random runs. (the benchmark is well approximated by LW with 20-million samples)

	Average KL Divergence
HMP-BN	0.0056
LW	0.0639

Table 4.2: Average KL-divergence Comparison in Testing GHM-2

4.3.3 Complexity of HMP-BN

Conditioning on every possible combination of instantiations of all interface nodes, HMP-BN computes the posterior distributions of hidden continuous variables given continuous evidence. So the complexity of the algorithm is highly dependent on the number of interface nodes and the size of the state space of the interface nodes. To test how HMP-BN performs in case of more interface nodes, we borrowed a network from the model ALARM [BSC89] shown in Figure 4.11, in which there are 37 nodes. Then we randomly select whether every node is discrete or continuous, subject to the requirement that continuous variables may not have any discrete children. In this experiment, the average number of interface nodes is around 12. HMP-BN still provided good estimates of the posterior distributions but it took much longer time than the one with only one interface node. In general, if we have m interface nodes $K_1, K_2, \ldots, K_m$ with number of states $n_1, n_2, \ldots, n_m$ respectively, the computational complexity of HMP-BN is approximately $\mathcal{O}(n_1 \times n_2 \times n_3 \ldots \times n_m)$. This implies that our algorithm is not scalable for large numbers of interface nodes. But our goal is not to propose an algorithm for all models (NP-hard in general). However, there are several ways to reduce the computational burden caused by the exponentially increased size of interface nodes. One method is to assume that interface nodes are independent of each other. Then we can propagate messages between continuous variables and individual interface node respectively. We have developed some initial ideas to compute generic messages between a continuous node and its discrete parents using loopy methods instead of network partition. Nevertheless, the detailed development of the corresponding algorithms is beyond the scope of this dissertation and will be discussed in our future research.

4.4 Summary

In this chapter, we developed a hybrid belief propagation algorithm for general Bayesian networks with mixed discrete and continuous variables. The algorithm first

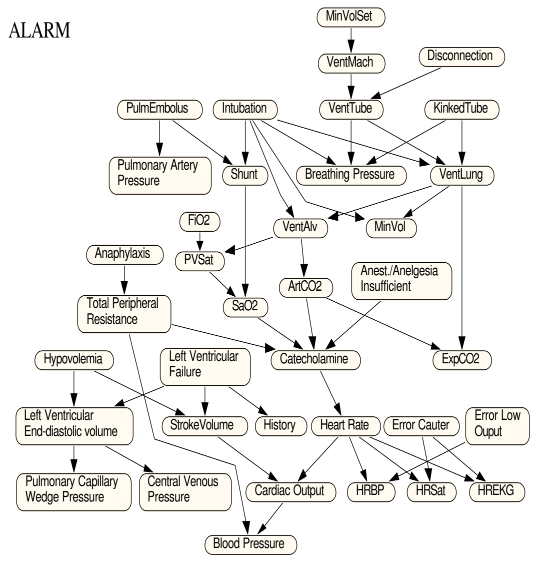


Figure 4.11: ALARM: a network constructed by medical expert for monitoring patients in intensive care.

partitions the network into discrete and continuous sub-networks by introducing interface nodes. Message passing is then applied to each network segment. The updated information is encoded as messages to be exchanged through the set of interface nodes. Finally, the algorithm integrates the separate messages from different network segments and computes the *a posteriori* distributions for all hidden nodes. Preliminary simulation results are encouraging and suggest that the algorithm will work well for hybrid Bayesian networks.

The main contribution of this algorithm is to provide a general framework for inference in hybrid models. Based on the principle of decomposing the complicated problem into smaller, affordable parts, we introduce the set of interface nodes to partition the network into disjoint either discrete or continuous sub-networks. Therefore, it is possible to apply exact inference algorithms such as Junction tree to some network segments and continuous variables are partitioned out so that we can handle them separately. For continuous network segment involving nonlinear and/or non-Gaussian variables, UMP-BN serves as an alternative inference algorithm.

Although the bottleneck of our algorithm is the size of interface nodes, we believe HMP-BN is a good option for nonlinear and/or non-Gaussian hybrid models since there are not many choices to deal with this case, especially given unlikely evidence.

Note that the focus of this chapter was on developing a unified message passing algorithm for general hybrid networks. While the algorithm works well to estimate means and variances for hidden continuous variables, the true posterior distributions may have multiple modes. In practice, it might be more important to know where the probability mass is than just knowing mean and variance. One idea is to utilize the messages computed in HMP-BN to obtain a good importance function and apply importance sampling to estimate the probability distributions [SC07c].

Chapter 5: Performance Evaluation for Bayesian Networks

In previous chapters, we focused on algorithm development for BN inference. Certainly, after building the model, it is very important to have an efficient way to compute the posterior probability distributions for hidden variables given observations. However, for realistic applications, it is also important to predict the model performance before applying it in inference algorithm. In many real-world decision problems, there is a hidden discrete variable, called target variable, with several categorical values representing the possible choices. One would like to know the probability that this target variable takes each of its possible values. Examples of such applications include target classification, medical diagnosis, marketing, etc. The common feature of these kinds of decision problems is the following: there is a hidden discrete target variable of interest, and there are observable continuous and/or discrete attributes. The values of observable features are caused directly or indirectly by the true value of the target variable. Given observations, the posterior probability distribution of the target variable is used to identify the causing class of the target variable. This chapter proposes an approximate analytical method to estimate the probability of correct classification for such decision models. The performance measure can help decision maker to understand the model's capability to identify the correct hypothesis without extensive simulation. The performance measure can also help the modeler to build and validate the model effectively.

5.1 Introduction

5.1.1 Performance Metric

First, we define a performance metric called 'probability of correct classification' (PCC). For a BN model of a decision problem, let us denote the target variable as T and a set of observable variables as $\mathbf{E}$ (evidence set). $\mathbf{E}$ consists of discrete attributes $\mathbf{E_d}$ and continuous attributes $\mathbf{E_c}$. Without loss of generality, we assume that T has n different discrete states. It will be very helpful for decision maker to know how well the model works. One indicator of performance is the probability that inferred target type is T_i given the true target type is T_j, i.e., $P(T_{inferred} = T_i | T_{true} = T_j)$. Obviously, given a true target type, better model will obtain higher probability that the inferred target type is being the same as the true target type. This indicator measures how well the model can identify the correct target. Now we are ready to give the formal definition of PCC:

Definition 5.1 (Probability of Correct Classification (PCC)). *For a target variable T with n discrete states in a Bayesian network model, probability of correct classification (PCC) is defined as the conditional probability mass function of target type given the true target type, denoted as $PCC = P(T_{inferred}|\ T_{true})$. PCC is a confusion table with entries defined as $PCC_{ij} = P(T_{inferred} = T_i|\ T_{true} = T_j)$.*

It is obvious that the diagonal elements of PCC table, $P(T_{inferred} = T_i|\ T_{true} = T_i)$, indicate the model's ability for correct identification.

5.1.2 Gaussian Mixture Model

In this section, let us deviate a little bit from BN inference and introduce an important statistical method called Gaussian mixture model (GMM). We will see how Gaussian mixture model is related to performance modeling for BN later in this chapter. GMM is very useful in general statistical analysis and density estimation. In fact, we already used it in computing the means and variances for continuous hidden variables in hybrid message passing in Chapter 4.

Any probability density function can be a basis density function to form a mixture. A mixture probability density function is a linear combination of basis density functions with all non-negative coefficients summing up to unity. In a mixture model, each basis density function is called a component. A k component Gaussian mixture model is a mixture density function consisting of k Gaussian density functions shown as below:

$$p(\mathbf{x}) = \sum_{i=1}^{i=k} p_i \mathcal{N}(\mathbf{x}; \bar{\mathbf{x}}_\mathbf{i}, \mathit{\Sigma}_\mathbf{i}) \tag{5.1}$$

where p_i is called mixing coefficient and

$$\sum_{i=1}^{i=k} p_i = 1, \quad 0 < p_i < 1 \tag{5.2}$$

Typically, the parameters of every Gaussian component $\mathcal{N}(\mathbf{x}; \bar{\mathbf{x}}_\mathbf{i}, \mathit{\Sigma}_\mathbf{i})$ are different. An example of 3-component two dimensional Gaussian mixture model is shown in Figure 5.1.

The mean and covariance of a k-component GMM shown in Equation 5.1, denoted

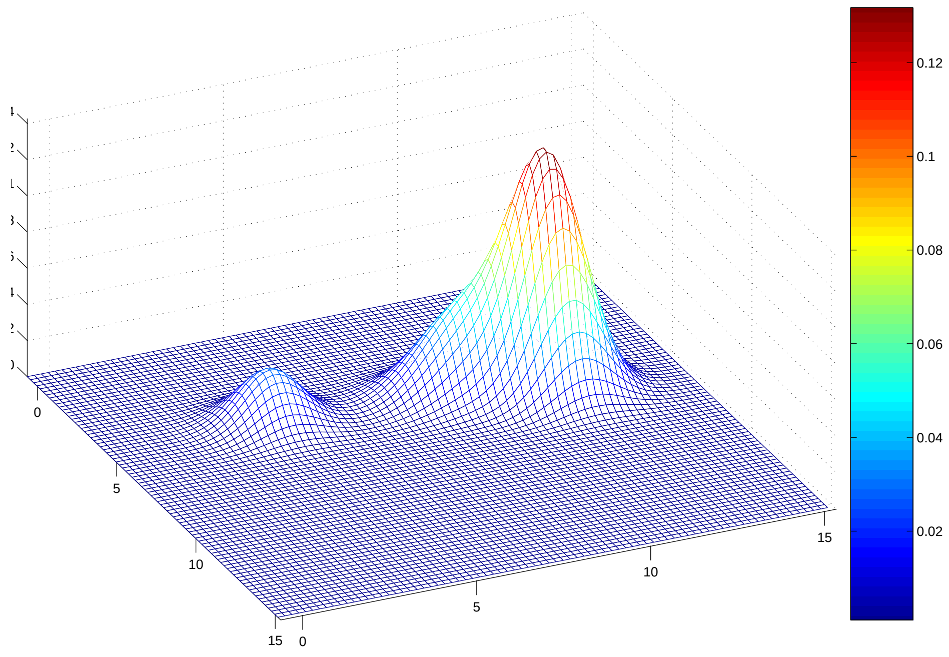


Figure 5.1: An example of two-dimension Gaussian mixture density with 3 components.

as $\bar{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$, are summarized as the following:

$$\bar{\mathbf{x}} = \sum_{i=1}^{i=k} p_i \bar{\mathbf{x}}_{\mathbf{i}} \tag{5.3}$$

$$\Sigma_{\mathbf{x}} = \sum_{i=1}^{i=k} p_i \Sigma_{\mathbf{i}} + \sum_{i=1}^{i=k} p_i \bar{\mathbf{x}}_{\mathbf{i}} \bar{\mathbf{x}}_{\mathbf{i}}' - \bar{\mathbf{x}}\bar{\mathbf{x}}' \tag{5.4}$$

Arbitrary probability density functions may have multiple modes and it is usually difficult to parameterize such density functions directly. GMM then plays an important role for density estimation. Theoretically, Gaussian mixture can approximate any density function arbitrarily closely provided it contains a sufficient number of components [AM79, MB88]. There have been many papers about learning a GMM from a data set [Dasg99, AK01, MP00]. One of the important learning issues is to determine the number of Gaussian components. A tradeoff must be made to balance the learning accuracy and the computational burden. *Expectation-maximization*, or known as EM, algorithm [Demp77] is one of the most popular GMM learning algorithms.

5.2 Analytical Performance Modeling

In this section, we show an analytical derivation to compute the probability of correct classification (PCC) defined in Section 5.1.1 for a hybrid BN decision model.

First of all,

$$\begin{aligned} PCC = P(T_{inferred}|\, T_{true}) &= \int_{\mathbf{E}} P(T_{inferred}, \mathbf{E}|\, T_{true})\, d\,\mathbf{E} \\ &= \int_{\mathbf{E}} P(T_{inferred}|\, \mathbf{E}, T_{true}) P(\mathbf{E}|\, T_{true})\, d\,\mathbf{E} \end{aligned} \tag{5.5}$$

The conditional distribution of the hidden target variable given the evidence is determined no matter what the hidden true state is. Therefore, we have

$$P(T_{inferred}|\, \mathbf{E}, T_{true}) = P(T_{inferred}|\, \mathbf{E}).$$

Then, Equation 5.5 becomes:

$$\begin{aligned} PCC &= \int_{\mathbf{E}} P(T_{inferred}|\, \mathbf{E}) P(\mathbf{E}|\, T_{true})\, d\,\mathbf{E} \\ &= \frac{1}{C} \int_{\mathbf{E}} P(\mathbf{E}|\, T_{inferred}) P(T_{inferred}) P(\mathbf{E}|\, T_{true})\, d\,\mathbf{E} \end{aligned} \tag{5.6}$$

where C is a normalization constant.

In general, $\mathbf{E}$ consists of discrete attributes $\mathbf{E_d}$ and continuous attributes $\mathbf{E_c}$. We will now derive the ij^{th} entry in the PCC matrix as the following (From the second equality, we drop the terms 'inferred' and 'true' and use T_i, T_j to represent different states):

$$\begin{aligned}
PCC_{i,j} &= P(T_{inferred} = T_i | \, T_{true} = T_j) \\
&= \int_{\mathbf{E}} P(T_i | \, \mathbf{E}) P(\mathbf{E} | \, T_j) \, d\,\mathbf{E} \\
&= \frac{1}{C} \int_{\mathbf{E}} P(\mathbf{E} | \, T_i) P(T_i) P(\mathbf{E} | \, T_j) \, d\,\mathbf{E} \\
&= \frac{1}{C} P(T_i) \int_{\mathbf{E_d}, \mathbf{E_c}} P(\mathbf{E_d}, \mathbf{E_c} | \, T_i) P(\mathbf{E_d}, \mathbf{E_c} | \, T_j) \, d\,\mathbf{E_d} d\,\mathbf{E_c} \\
&= \frac{1}{C} P(T_i) \int_{\mathbf{E_c}} P(\mathbf{E_c} | \mathbf{E_d}, T_i) P(\mathbf{E_c} | \mathbf{E_d}, T_j) \cdot \\
&\qquad \sum_{\mathbf{E_d}} P(\mathbf{E_d} | \, T_i) P(\mathbf{E_d} | \, T_j) \, d\,\mathbf{E_c} \qquad (5.7)
\end{aligned}$$

where C is a normalization constant and $P(T_i)$ is the prior probability of the i^{th} state of T.

Equation 5.7 is the optimal analytical solution for PCC. However, it is not trivial to derive $P(\mathbf{E_d} | \, T)$ and $P(\mathbf{E_c} | \, \mathbf{E_d}, T)$ because typically there are intermediate variables between T and evidence nodes $\mathbf{E_d}, \mathbf{E_c}$. Another key factor that makes the computation difficult is the heterogeneity (discrete and continuous observations mixed together). In the case of pure discrete observations or by discretizing continuous attributes, some efficient ways were proposed to evaluate performance confusion matrix

[EC07].

In general, by forward sampling, we could approximate $P(\mathbf{E_d}|\, T)$ and $P(\mathbf{E_c}|\, \mathbf{E_d}, T)$. But the combination of $\mathbf{E_d}$ and T make the computational complexity exponential. It is relatively easy if we can assume $\mathbf{E_d}, \mathbf{E_c}$ are independent; then $P(\mathbf{E_d}, \mathbf{E_c}|\, T) = P(\mathbf{E_d}|\, T)P(\mathbf{E_c}|\, T)$ and $P(\mathbf{E_d}|\, T), P(\mathbf{E_c}|\, T)$ could be estimated respectively. But this is not always the case in most applications.

Since it is too difficult to compute the PCC matrix with hybrid observations, here we solve a simplified problem by assuming all observations are continuous. Then Equation 5.7 is reduced to:

$$PCC_{i,j} = \frac{1}{C} P(T_i) \int_{\mathbf{E}} P(\mathbf{E}|\, T_i) P(\mathbf{E}|\, T_j)\, d\,\mathbf{E} \tag{5.8}$$

If we can approximate the conditional probability distribution $P(\mathbf{E}|\, T)$ as a Gaussian distribution with sufficient accuracy, then the above equation has a closed-form analytical solution, because integral of a product of two Gaussian density functions is also a Gaussian density function. Namely, suppose

$$P(\mathbf{E}|\, T_i) \sim \mathcal{N}(u_i, P_i), \qquad P(\mathbf{E}|\, T_j) \sim \mathcal{N}(u_j, P_j)$$

where u_i, u_j are mean vectors, P_i, P_j are covariance matrices. Then Equation 5.8

becomes:

$$\begin{aligned} PCC_{i,j} &= \frac{1}{C}P(T_i)\int_{\mathbf{E}}\mathcal{N}(u_i,P_i)\mathcal{N}(u_j,P_j)\,d\,\mathbf{E} \\ &= \frac{1}{C}P(T_i)\frac{1}{2\pi}|P_iP_j|^{-\frac{1}{2}}\int_{\mathbf{E}}e^{-\frac{1}{2}\left[(\mathbf{E}-u_i)'{P_i}^{-1}(\mathbf{E}-u_i)-(\mathbf{E}-u_j)'{P_j}^{-1}(\mathbf{E}-u_j)\right]}\,d\,\mathbf{E} \\ &= \frac{1}{C}P(T_i)|2\pi(P_i+P_j)|^{-\frac{1}{2}}e^{-\frac{1}{2}(u_i-u_j)'(P_i+P_j)^{-1}(u_i-u_j)} \qquad (5.9) \end{aligned}$$

In reality, $P(\mathbf{E}|T)$ could be a complicated multi-dimensional mixture distribution with possibly multiple modes. So it is usually not good enough to approximate $P(\mathbf{E}|\ T)$ with one single Gaussian distribution. We know that a Gaussian mixture model with appropriate number of components can approximate arbitrary probability distribution to any accuracy. Let us assume $P(\mathbf{E}|\ T_i)$ and $P(\mathbf{E}|\ T_j)$ can be estimated by m-component and n-component GMMs respectively, namely:

$$P(\mathbf{E}|\ T_i) \sim \sum_{k=1}^{k=m} a_k\mathcal{N}(u_{ik},P_{ik}), \quad P(\mathbf{E}|\ T_j) \sim \sum_{l=1}^{l=n} b_l\mathcal{N}(u_{jl},P_{jl}) \qquad (5.10)$$

where a_k, b_l are mixing coefficients, u_{ik}, P_{ik} are mean vectors and covariance matrices of the k^{th} Gaussian component of a GMM for $P(\mathbf{E}|\ T_i)$; similarly, u_{jl}, P_{jl} are mean vectors and covariance matrices of the l^{th} Gaussian component of a GMM for $P(\mathbf{E}|\ T_j)$.

Then,

$$
\begin{aligned}
PCC_{i,j} &= \frac{1}{C}P(T_i)\int_{\mathbf{E}}\sum_{k=1}^{k=m} a_k\mathcal{N}(u_{ik},P_{ik})\sum_{l=1}^{l=n} b_l\mathcal{N}(u_{jl},P_{jl})\, d\,\mathbf{E} \\
&= \frac{1}{C}P(T_i)\sum_{k=1}^{k=m} a_k\sum_{l=1}^{l=n} b_l\int_{\mathbf{E}}\mathcal{N}(u_{ik},P_{ik})\mathcal{N}(u_{jl},P_{jl})\, d\,\mathbf{E} \\
&= \frac{1}{C}P(T_i)\sum_{k=1}^{k=m}\sum_{l=1}^{l=n}\{a_k b_l|2\pi(P_{ik}+P_{jl})|^{-\frac{1}{2}}e^{-\frac{1}{2}(u_{ik}-u_{jl})'(P_{ik}+P_{jl})^{-1}(u_{ik}-u_{jl})}\} \\
&\qquad e^{-\frac{1}{2}(u_{ik}-u_{jl})'(P_{ik}+P_{jl})^{-1}(u_{ik}-u_{jl})}\} \qquad (5.11)
\end{aligned}
$$

Therefore, as derived above, we can obtain analytical solutions for the confusion matrix PCC with the approximation coming from GMM learning. Obviously, additional computational cost is paid to learn the corresponding GMMs and control the accuracy by determining the number of Gaussian components. We implement the learning process by sampling and estimation. First, we generate random samples of $\mathbf{E}$ given every state of T by forward sampling, then we apply EM algorithm to estimate the corresponding GMMs by using a MATLAB toolbox called Netlab [Nab02]. Finally, according to Equation 5.11, we compute PCC confusion matrix analytically.

5.3 Numerical Evaluation

To evaluate this approximate analytical method of performance modeling, we use a 16-node hybrid Bayesian network as the experimental model shown in Figure 5.2. We have 6 observable attributes in this model and they are all continuous. Note that 'TT'(target type) is the discrete target variable of interest. We assume 'TT' has 10 discrete states with uniform prior distribution. Other nodes in this model are linear Gaussian variables. So the experiment model is a CLG (for definition and other details of CLG, see Chapter 2). For benchmark inference results, we use clique tree algorithm [Lau92, LJ01] to compute the exact posterior distributions.[1] Therefore, we can conduct simulations to compare the model performance with analytical prediction computed by Equation 5.11.

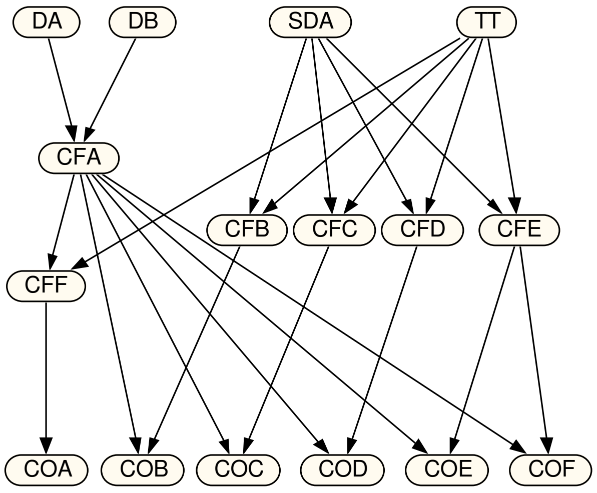


Figure 5.2: Synthetic hybrid model of target identification: 'TT' is the discrete target variable; 'COA','COB','COC','COD','COE' and 'COF' are observable continuous attributes.

We evaluate *PCC* using two different methods. The first is to compute *PCC* analytically by using Equation 5.11. In this step, no inference algorithm is needed because we do not need to know the posterior probability distributions. However, forward sampling is conducted to generate random samples of observations given

[1]Note that since this experimental model is a CLG, clique tree algorithm could be applied to conduct exact inference.

every state of the target variable T. The complexity of forward sampling is linear and it is straightforward to implement. From the random sample, we apply EM algorithm to learn GMMs for estimating $P(\mathbf{E}|T)$. In our experiments, we use 3-component GMMs and 10-component GMMs respectively. After we have learned the GMMs to approximate the conditional distribution of evidence given target type, we can compute the PCC matrix analytically using Equation 5.11.

The second method is to compute the PCC matrix by extensive simulation. In this approach, we need to invoke an inference algorithm to compute the posterior distribution of target type T given random sample of observations for each simulation trial. So the computational complexity will depend on the complexity of the inference algorithm too. In particular, given each target type T_i, we generate $1,000$ random samples of observations. Then given each set of observations, we use clique tree algorithm to compute the exact posterior distribution of T. These posterior probabilities are then averaged over these $1,000$ simulation trials to obtain the i^{th} row of the PCC matrix, $P(T_{inferred}|T_i)$. This process takes significant more time than analytical method does and it is also an approximate performance estimation because of two sources: 1. simulation; 2. the inference algorithm itself may be approximate.

We compare PCC matrices provided by these two different methods of performance modeling. Especially, it is more interesting to compare the diagonal elements of PCC matrix, $P(T_i|T_i), i = 1, 2, ..., 10$, because they indicate the model's ability to identify the correct target type. As illustrated in Figure 5.3, the diagonals of PCC matrices are plotted for three cases respectively. The black line with circle points represents PCC diagonal provided by 10×1000 simulation runs where within each run clique tree algorithm is used to compute the posterior distribution. The blue line with square points represents PCC diagonal provided by analytical method using 3-component GMM. The red line with diamond points represents PCC diagonal provided by analytical method using 10-component GMM. From this diagram, we see that the predictions of the analytical method with GMMs agree very closely to predictions obtained by extensive simulation. It seems that 3-component GMM is good enough in this case. In terms of time, the analytical method is certainly more efficient in evaluating model performance.

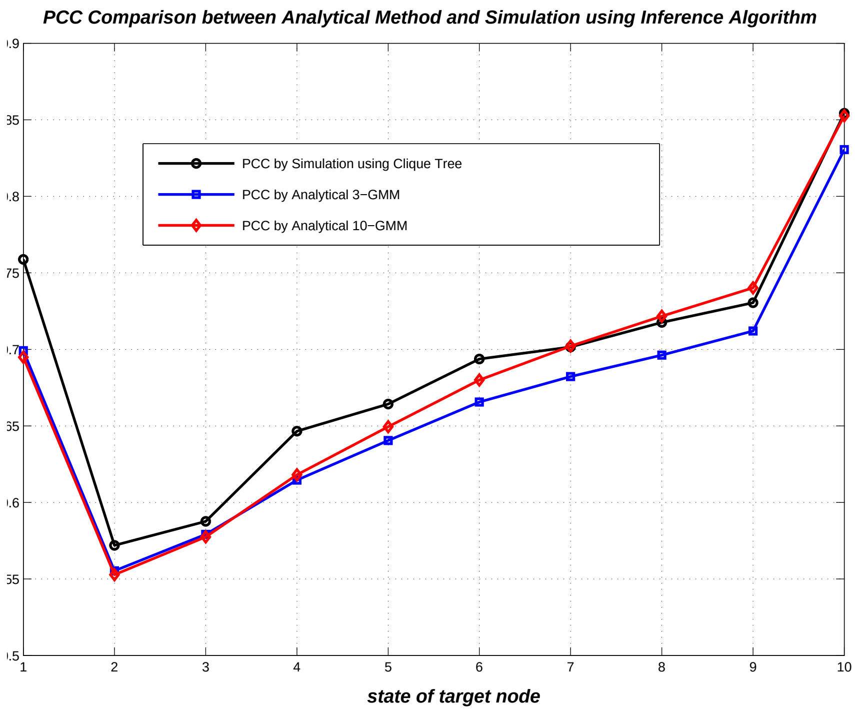


Figure 5.3: PCC comparison between analytical method with GMMs and simulation using inference algorithm.

Furthermore, note that given a state of target node, $PCC(i, i)$ explicitly indicates the probability of correctly identifying the i^{th} true state. For example, it shows from Figure 5.3, that the probability of correctly identifying the 7^{th} state given the true state is the 7^{th} is about 0.7. Therefore, by estimating PCC confusion table, we can understand the model prediction performance. This is very important in decision making and model construction and evaluation.

5.4 Summary

Bayesian Network as a decision support tool has been popular for several decades. In modeling decision problems, BNs usually have a discrete variable of the decision maker's interest, called target variable T, and a set of observable attributes, called evidence $\mathbf{E}$. The target node in a BN model is a hidden variable. One way to infer the value of the target node is to compute its posterior distribution given the observations of evidence nodes. This is commonly referred to as Bayesian inference. It is desirable to have a way to evaluate the inference performance for a BN model before it is implemented. This will help in designing and analyzing a Bayesian classification system. Take a senor fusion network as an example, understanding the model performance can help to deploy the sensors accordingly.

In this chapter, we first present a performance metric called probability of correct classification (PCC). We then propose an approximate analytical method to model the inference performance for a BN model. This analytical method uses Gaussian mixture model to estimate the conditional probability distribution of evidence given target type $P(\mathbf{E}|T)$. The only approximation of this method is from GMM learning. And the main computational tasks are random sample generation by forward sampling and GMM learning by EM algorithm. This analytical GMM performance modeling method does not need to apply any inference algorithm, while oppositely, performance predictions by extensive simulation must apply inference algorithms to compute the posterior distributions. Regarding the accuracy, the numerical experiment results demonstrate promising performance.

When hybrid evidence is involved (continuous and discrete observations mixed together), so far there is no efficient means to predict inference performance in the literature. It would be interesting to explore some possible approaches such as continuzing discrete attributes or discretizing continuous attributes. Moreover, for dynamic BN, things become even more complicated. We expect that some filter-based methods may be used for DBN performance modeling.

Chapter 6: Conclusion

6.1 Dissertation Summary

Although probabilistic inference using Bayesian networks is NP-hard in general, the essential issue is the tradeoff of computational complexity and performance accuracy. In many real-world applications, which are usually not close to the theoretically demonstrated worse cases, inference is tractable by either exact or approximate methods. In particular, efficient inference algorithms should take advantages of the network topology and specific types of variables in the model.

Since there have been many well developed algorithms for pure discrete BNs and the simplest hybrid model-CLGs, this dissertation concentrates on inference for hybrid BNs with arbitrary relationships and probability distributions. Our major contributions are the development of approximate inference algorithms and analytical performance bound estimation for hybrid BN models.

Three main categories for approximate BN inference are: (1) stochastic sampling; (2) model simplification; (3) loopy belief propagation (LBP). The first one is an ultimate option when all other inference methods fail. It is well known that the efficiency of forward stochastic sampling algorithms depends highly on the chosen sampling probability distribution function, the so-called importance function. Although there are some methods proposed to find good importance functions for pure discrete BNs such as AIS-BN and EPIS-BN, it is still an open issue for hybrid BNs.

The methodologies we used in this dissertation for efficient hybrid BN inference are based on loopy belief propagation, also called message passing. Researchers have reported surprisingly good results of loopy message passing for inference in pure discrete BNs and other applications such as decoding, visual tracking as well. We proposed an extension of loopy message passing in hybrid BN. First of all, we used the first two moments of continuous variable to represent messages in continuous case. Then a suite of algorithms was developed for messages propagation between arbitrary continuous variables and messages integrations between discrete and continuous variables. These approaches formulate a general framework of approximate inference for hybrid Bayesian networks. Compared with the model-free stochastic sampling method - likelihood weighting, our algorithms achieve orders of magnitude improvement in the majority of numerical test cases.

On the other hand, the ability to correctly identify the state of discrete variable in BN models is very important in classification problems. In Chapter 5, we defined a model performance measure as probability of correct classification (PCC). PCC is a confusion table in which each row is the posterior probability mass function of a discrete target variable given one of its states. To compute PCC efficiently, we proposed an approximate analytical estimator of performance bound for hybrid BNs with discrete target variable and continuous observations. This analytical performance modeling method does not require the execution of an inference algorithm, instead, it integrates Gaussian mixture models for density estimations. Therefore, extensive simulation with possibly complicated inference algorithm is avoided. We know that PCC depends on the constructed model including network structure and

CPDs. So the model performance bound can help the modeler to adjust the model accordingly and more importantly, it can help decision maker to understand the model prediction power for practical applications.

In summary, this dissertation aims at the difficult problems with nonlinearity and non-Gaussian in hybrid Bayesian network inference. We proposed a message-passing-based framework to compute the posterior probability distribution for hybrid BNs. Algorithms developed under this framework are general enough to deal with hybrid model with any functional relationships and any type of variables. we believe these algorithms can serve as good alternatives when other methods are either not applicable or not sufficiently efficient.

6.2 Future Work

The inference problem for hybrid BN model is difficult enough so that there is room for much work to be done. There are many potential interesting research directions that build upon this dissertation. As follows, we outline a few research topics for future work:

- Although surprisingly good results of convergence have been reported for loopy belief propagation empirically, there lack a theoretical explanation or insight for these good experimental results. There are cases in which loopy belief propagation does not converge but exhibits oscillations between values that have very little correlation with the correct marginal distributions. we believe that theoretical derivation of loopy propagation is an important area for further research. Exploratory idea and new theoretical insight on loopy propagation can have significant impact on developing efficient inference methods.

- In UMP-BN, we assume that the parents of continuous variable are independent and so establish the diagonal covariance matrix accordingly. This is not true in general. One idea for future research is to estimate the correlations between the parent variables and build the corresponding full covariance matrix. It would be interesting to explore the tradeoff between performance and computational costs for the resulting algorithm.

- Our algorithms return estimates of the first two moments for continuous variables. However, sometimes, it is more important to know the whole distributions instead of just two moments. One approach for future study is to use Gaussian mixture with more than one component and develop the corresponding message propagation and integration algorithms. Also, to further improve the performance, we always can apply the estimated distributions as the importance function for additional importance sampling.

- The hybrid BNs we can deal with now is the CHM, in which there is no discrete children for any continuous variable. Researchers have proposed some methods and conducted investigations for hybrid BN models without this restriction [Lern02, CS06, YD06]. In this dissertation, the main reason for making this restriction is so that we could partition the original hybrid BNs into either pure discrete or pure continuous network segments. In general, if one could develop efficient methods to propagate messages directly between different types of variables, this could lead to the development of message-passing-based algorithms for general hybrid BN models that relax this assumption.

- Probabilistic inference for dynamic Bayesian networks is typically conducted sequentially and recursively. First of all, we need to have an efficient inference algorithm for static BNs, which is one-time-slice of DBNs. Then under some rollup mechanism or filter-typed framework, current inference results must be propagated and integrated for the next time slice when new observations are available. We expect that the algorithms developed in this dissertation could be adapted for DBN inference.

Probabilistic reasoning is one of the most important issues in the fields of statistics, operations research, and artificial intelligence. This dissertation proposed methods to address computational difficulty caused by nonlinearity and non-Gaussian for inference in hybrid Bayesian networks. We developed methodologies to approximate the arbitrary distribution by Gaussian and achieve good estimates of the first two moments for continuous variables in hybrid model. We believe that these inference algorithms and analytical performance modeling developed in this dissertation significantly extend the applicability, efficiency and understanding of approximate inference for Bayesian networks.

Bibliography

Bibliography

[AM79] Brian D.O. Anderson and J.B. Moore. *Optimal Filtering.* Prentice-Hall, Englewood Cliffs, NJ, 1979.

[AMG02] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. *A Tutorial on Particle Filters for on-line nonlinear/non-Gaussian Bayesian tracking.* IEEE Trans. Signal Process., vol. 50, no. 2, pp. 174–188, Feb. 2002.

[AK01] S. Arora and R. Kannan. *Learning Mixtures of Arbitrary Gaussians.* Symposium on Theory of Computing (STOC), 2001.

[BD06] Bozhena Bidyuk, Rina Dechter. *Cutset Sampling for Bayesian Networks.* In Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06), Arlington, Virginia, 2006.

[BK98] Xavier Boyen and Daphne Koller. *Tractable inference for complex stochastic processes.* In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pages 33–42, Madison, WI, 1998.

[BLK01] Yaakov Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*, Wiley, 2001.

[BSC89] I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. *The Alarm Monitoring System: A Case Study with Two Probabilistic Inference techniques for Belief Networks.* In Proceedings of 2^{nd} European Conference on AI and Medicine, 1989.

[CACM95] *Communications of the ACM*, Special Issue on Real-World Applications of Bayesian Networks, D. Heckerman, A. Mamdani, M. WEllman (editors) 38(3), March 1995.

[CD00] Jian Cheng and Marek J. Druzdzel. *AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks.* Journal of Artificial Intelligence Research (JAIR), 13:155-188, 2000.

[CDLS99] Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Statistics for Engineering and Information Sciences. New York: Springer-Verlag, 1999.

[Cheng01] Jian Cheng, PhD dissertation, *Efficient Stochastic Sampling Algorithms for Bayesian networks*, School of Information Sciences, University of Pittsburgh, 2001.

[CF95] K. Chang, R. Fung. *Symbolic probabilistic inference with both discrete and continuous variables.* IEEE Transactions on Systems, Man, and Cybernetics, vol. 25, pp. 910 - 916, 1995.

[Co84] G.F. Cooper, *Nestor: A computer-based medical diagnosis aid that integrates causal and probabilistic knowledge.* Ph.D. dissertation, Department of Computer Science, Stanford University, 1984.

[Co90] G. F. Cooper. *The computational complexity of probabilistic inference using Bayesian belief networks.* Artificial Intelligence, vol. 42, pp. 393-405, 1990.

[CS04] KC Chang and Wei Sun. *Performance Modeling for Bayesian Networks.* In Proceedings of SPIE Conference, Volume 5429, Orlando, 2004.

[CS06] Barry R. Cobb and Prakash P. Shenoy. *Inference in Hybrid Bayesian Networks with Mixtures of Truncated Exponentials.* International Journal of Approximate ReasoningVolume 41, Issue 3, , April 2006, Pages 257-286.

[Dasg99] S. Dasgupta. *Learning Mixtures of Gaussians.* Proceedings of Symposium on Foundations of Computer Science (FOCS), 1999.

[DeGr70] Morris H. de Groot. *Optimal Statistical Decisions.* McGraw-Hill and Company (1970).

[Demp77] Dempster, A.P. and N.M. Larid, D.B. Rubin. *Maximum Likelihood from Imcomplete data set via the EM algorithm.* Journal of the Royal Statistical Society. B 39(1), p1-38, 1977.

[DFG01a] A. Doucet, J. Freitas, and N. Gordon. *A introduction to sequential Monte Carlo methods in practice.* In *Sequential Monte Carlo Methods in Practice*, A. Doucet, J. Freitas, and N. Gordon, Eds., chapter 1. Springer-Verlag, New York, 2001.

[DFG01b] A. Doucet, J. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice.* Springer-Verlag, New York, 2001.

[DL93] P. Dagum and M. Luby. *Approximating probabilistic inference in Bayesian belief networks is NP–hard.*Artificial Intelligence, vol. 60, pp. 141-153, 1993.

[dMer04] Rudolph van der Merwe. PhD thesis, *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models.* Oregon Health & Science University. 2004.

[Dun75] Duncan, O. *Introduction to Structural Equation Models.* New York: Academic Press, 1975.

[EC07] Eswar Sivaraman and KC Chang. *Performance Evaluation of Multi-Sensor Classification Systems.* IEEE Transactions on Aerospace and Electronic Systems, November, 2007.

[FC89] R. Fung and K. C. Chang. *Weighting and integrating evidence for stochastic simulation in Bayesian networks.* In Uncertainty in Artificial Intelligence 5,pages 209-219, New York, N. Y., 1989. Elsevier Science Publishing Company, Inc., 1989.

[GB01] W. Gilks and C. Berzuini. *Following a moving target–Monte Carlo Inference for Bayesian Dynamic Models.* Journal of Royal Statistic Society B, vol. 63, pp. 127–146, 2001.

[GG84] S. Geman and D. Geman. *Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 6:721-741, 1984.

[GH02] H. Guo and W. Hsu, *A survey of algorithms for real-time Bayesian network inference*, AAAI/KDD/UAI–2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems, Edmonton, Alberta, Canada, 2002.

[GRS96] W.R. Gilks, S. Richardson, and David Spiegelhater. *Markov Chain Monte Carlo in Practice.* Chapman & Hall/CRC, 1996.

[Gol72] Goldberger, A. *Structural equation models in the social sciences.* Econometrica, 40: 979-1001, 1972.

[Haa43] Haavelmo, T. *The statistical implications of a system of simultaneous equations.* Econometrica, 11: 1-12, 1943.

[Haddawy99] P. Haddawy. *An Overview of Some Recent Developments in Bayesian Problem-Solving Techniques.* AI Magazine 20(2) p11-19, 1999.

[Hast70] W.K. Hastings. *Monte Carlo Sampling Methods Using Markov Chains and Their Applications.* Biometrika, 57(1):97-109, 1970.

[Hay98] Simon Haykin. *Neural Networks: A Comprehensive Foundation*, Prentice Hall PTR, NJ, isbn-0132733501, 1998.

[HD96] Cecil Huang and Adnan Darwiche. *Inference in belief networks: A procedural guide*, *International Journal of Approximate Reasoning*, 15:225–263, 1996.

[Hen88] Max Henrion. *Propagation of uncertainty in Bayesian networks by probabilistic logic sampling.* In Uncertainty in Artificial Intelligence, vol. 2. Elvesier, 1988, pp. 149–163.

[HM81] Howard, R.A. and Matheson, J.E. *Influence Diagrams*, in Principles and Applications of Decision Analysis, Manlo Park CA, Strategic Decision Group. 1981.

[HM05] Howard, R.A. and Matheson, J.E. *Influence Diagram Retrospective*, INFORM Decision Analysis, Vol.2, No.3, pp.144-147, Sep., 2005.

[Jensen96] F.V. Jensen. *An Introduction to Bayesian Networks.* Springer-Verlag, New York, 1996.

[JMM96] Jain, A.K.; Jianchang Mao; Mohiuddin, K.M., "Artificial neural networks: a tutorial," Computer , vol.29, no.3, pp.31-44, Mar 1996.

[JU96] S. J. Julier and J. K. Uhlmann. *A General Method for Approximating Nonlinear Transformations of Probability Distributions.* Technical report, RRG, Dept. of Engineering Science, University of Oxford, Nov 1996.

[JU97] S.J. Julier and J.K. Uhlmann. *A New Extension of the Kalman Filter to Nonlinear Systems.* In The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, (Orlando, Florida), 1997.

[Julier02] Julier, S. J. *The Scaled Unscented Transformation.* In Proceedings of the American Control Conference, vol. 6, pp. 4555C4559. 2002.

[Kal60] Kalman, R. E. *A new approach to linear filtering and prediction problems.* Trans. ASME, Series D, Journal of Basis Engineering 82, 35–45, 1960.

[KF98] D. Koller and R. Fratkina, *Using Learning for Approximation in Stochastic Processes.* In Proc. of the 15th International Conference on Machine Learning (ICML–98), pages 287–295, Madison, Wisconsin, July 1998.

[KL99] Dephne Koller, Uri Lerner, and Dragomir Angelov. *A General Algorithm for Approximate Inference and Its Application to Hybrid Bayes Nets*. In Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99), pages 324-333, Stockholm, Sweden, July 30th - August 1st, 1999.

[Koo50] Koopmans, T. *When is an equation system complete for statistical purposes?*. In T. Koopmans (Ed.), Statistical inference in Dynamic Economic Models (Cowles Commission Monograph No. 10). New York: Wiley.1950.

[Lau92] Steffen L. Lauritzen. *Propagation of probabilities, means, and variances in mixed graphical association models*. JASA, 87(420):1089–1108, 1992.

[Lern02] Uri N. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems*, Ph.D. Thesis, Stanford University, October 2002.

[LJ01] S. L. Lauritzen and F. Jensen. *Stable local computations with conditional Gaussian distributions. Statistics and Computing*, vol.11, no.2, pp191–203, April 2001.

[LM-etc02] U. Lerner, B.Moses, M.Scott, S. Mcllraith and D. Koller. *Monitoring a complex physical system using a hybrid dynamic Bayes net*. In Proceedings of the 18^{th} Annual Conference on Uncertainty in AI (UAI), pages 301-310, 2002.

[LS88] S.L. Lauritzen and D.J. Spiegelhalter, *Local Computations with Probabilities on Graphical Structures and Their Applications to Expert Systems*, Proceedings of the Royal Statistical Society, Series B., 50, 157-224, 1988.

[MB88] McLachlan, G.J. and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. New York: Marcel Dekker, 1988.

[MDFW00] Rudolph van der Merwe, A. Doucet, N. de Freitas, and E. Wan. *The Unscented Particle Filter*. Tech. Rep., Department of engineering, University of Cambridge CB2 1PZ Cambridge, 2000.

[MP00] G. McLachlan, D. Peel. *Finite Mixture Models*. Wiley, 2000.

[MRR+53] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. *Equations of State Calculations by Fast Computing Machines*. Journal of Chemical Physics, 21(6):1087-1092, 1953.

[MRS01] S. Moral, R. Rumi, A. Salmeron (2001).*Mixtures of truncated exponentials in hybrid Bayesian networks.* Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty. Lecture Notes in Artificial Intelligence 2143, 156-167. Berling, Springer-Verlag.

[MRS02] S. Moral, R. Rumí, A. Salmerón. *Estimating mixtures of truncated exponentials from data.* In: J.A. Gámez, A. Salmerón (Eds.), Proceedings of the First European Workshop on Probabilistic Graphical Models (PGMC02), Cuenca, Spain, 2002, pp. 135C143.

[Mur98] Kevin Murphy. *Inference and learning in hybrid bayesian networks.* Technical Report CSD-98-990, Department of Computer Science, U.C. Berkeley, 1998.

[Mur01] Kevin Murphy. *The Bayes Net Toolbox for Matlab.* In Computing Science and Statistics: Proceedings of the Interface, Volume 33, 2001.

[Mur02] Kevin Murphy, PhD Thesis, *Dynamic Bayesian Networks: Representation, Inference and Learning.* UC Berkeley, 2002.

[MWJ99] K. Murphy, Y. Weiss, and M. Jordan. *Loopy belief propagation for approximate inference:an empirical study.* In UAI, 1999.

[Nab02] Ian T. Nabney. *NETLAB: Algorithms for Pattern Recognition.* Springer, 2002.

[Ne90] R. E. Neapolitan. *Probabilistic Reasoning in Expert Systems.* John Wiley & Sons, NY, 1990.

[Neal93] Radford M. Neal. *Probabilistic Inference using Markov Chain Monte Carlo Methods.* Technical Report CRG-TR-93-1, University of Toronto, 1993.

[Pearl82] Judea Pearl. *Reverend bayes on inference engines: A distributed hierarchical approach.* In David Waltz, editor, Proceedings of the National Conference on Artificial Intelligence, pages 133–136, Pittsburgh, PA, August 1982. AAAI Press.

[Pearl86] J. Pearl, *Fusion, propagation and structuring in belief networks*, Artificial Intelligence, vol. 29, pp. 241-288, 1986.

[Pearl87] Judea Pearl. *Evidential reasoning using stochastic simulation.* Artificial Intelligence, pages 245–257, 32, 1987.

[Pearl88] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kauffman, San Mateo, 1988.

[Pearl05] J. Pearl, *Influence Diagrams–Historical and Personal Perspectives.* INFORM Decision Analysis, Vol.2, No.4, pp.232-234, Dec., 2005.

[Poland94] Poland, W.B. 1994. *Mixture of Gaussians and Minimum Relative Entropy Techniques for Modeling Continuous distributions.* PhD thesis, Department of Engineering-Economic Systems, Stanford University.

[PS91] Mark A. Peot and Ross D. Shachter, *Fusion and propagation with multiple observations in belief networks.* Artificial Intelligence, vol.48, pp.299-318, 1991.

[PS99] M.K. Pitt and N. Shephard. *Filtering via Simulation: Auxiliary Particle Filters.* Journal of the American Statistical Association. Vol 94(446), pp 590–599, 1999.

[Rubin81] Rubinstein, R. Y. *Simulation and the Monte Carlo Method.* John Wiley & Sons, 1981.

[SC05] Wei Sun and K.C. Chang, *Probabilistic Inference using Linear Gaussian Importance Sampling for General Hybrid Bayesian Networks.* SPIE Defense and Security Symposium, Vol.# 5809, Orlando, April 2005.

[SC07a] Wei Sun and KC Chang. *Unscented Message Passing for Arbitrary Continuous Bayesian Networks.* Proceedings of the 22nd AAAI Conference on Artificial Intelligence, Vancouver, Canada, 2007.

[SC07b] Wei Sun and KC Chang. *Hybrid Message Passing for General Mixed Bayesian Networks.* Proceedings of the 10th International Conference on Information Fusion, Quebec, Canada, 2007.

[SC07c] Wei Sun and KC Chang. *Hybrid Loopy Importance Sampling for Bayesian Networks.* To be submitted.

[SC08a] Wei Sun and KC Chang. *Convergence Study of Message Passing in Arbitrary Continuous Bayesian Networks.* Submitted to SPIE Conference, Orlando, March, 2008.

[SD90] R. D. Shachter, B. D'Ambrosio, and B. D. Del Favero. *Symbolic Probabilistic Inference in Belief Networks*, Proc. 8th National Conference on Artificial Intelligence, MIT Press, Boston, pp. 126131, 1990.

[Sh86] Ross D. Shachter. *Intelligent probabilistic inference.* In J. F. Lemmer and L. N. Kanal, editors, Uncertainty in Artificial Intelligence, pages 371–382. North Holland, Amsterdam, 1986.

[Sh90] Ross D. Shachter. *Evidence Absorption and Propagation through Evidence Reversals.* In M. Henrion,R. D. Shachter,J. F. Lemmer, & L. N. Kanal (Eds.), Uncertainty in Artificial Intelligence 5 (pp. 173-190), 1990.

[Shenoy06] Prakash Shenoy. *Inference in Hybrid Bayesian Networks Using Mixtures of Gaussians.* Proceedings of the 22^{nd} Annual Conference on Uncertainty in Artificial Intelligence (UAI), 2006.

[SP90] R. D. Shachter and M. A. Peot, *Simulation approaches to general probabilistic inference on belief networks.* In Proc. of the Conf. on Uncertainty in AI,volume 5, 1990.

[SS90] P. P. Shenoy and G. R. Shafer. *Axioms for probability and belief-function propagation.* In Proc. UAI, volume 4, pages 169-198, 1990.

[TAW02] M. Takikawa, B. DAmbrosio, and E. Wright. *Real-time inference with large-scale temporal bayes nets.* In Proc. of the Conf. on Uncertainty in AI, 2002.

[WF99] Weiss, Y. and W. T. Freeman. *Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology.* Technical Report, UCB.CSD-99-1046, Berkeley Computer Science Dept. 1999.

[Wri21] Wright, Sewall. *Correlation and causation.* Journal of Agricultural Research 20: 557-85, 1921.

[Wri23] Wright, Sewall. *The theory of path coefficients: A reply to Niles criticism.* Genetics, 8: 239-55, 1923.

[Wri34] Wright, Sewall. *The Method of Path Coefficients.* Ann Math Statist, 5, 161-215, 1934.

[YD03] C. Yuan, M. J. Druzdzel. *An Importance Sampling Algorithm Based on Evidence Pre-propagation.* In Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence, pages 624-631, August 2003.

[YD06] C. Yuan and M. J. Druzdzel. *Hybrid Loopy Belief Propagation.* Proceedings of the third European Workshop on Probabilistic Graphical Models (PGM-06), pages 317-324, 2006.

[YD07] Changhe Yuan, and Marek J. Druzdzel. *Generalized Evidence Pre-propagated Importance Sampling for Hybrid Bayesian Networks.* In Proceedings of the Twenty-second National Conference on Artificial Intelligence (AAAI-07). Vancouver, Canada.

[ZP94] N. L. Zhang and D. Poole. *A simple approach to Bayesian network computations.* In Proc. of the Tenth Canadian Conference on Artificial Intelligence, pages 171–178, 1994.

Curriculum Vitae

Wei Sun received his Bachelor of Science in Electrical Engineering in 1991 from Zhejiang University, Hangzhou city, China. He was employed as an electrical engineer in Guizhou No.1 Power Plant Construction Company in China for several years before he came to the United States for his graduate studies in 2001. Wei Sun obtained his Master of Science in Operations Research from George Mason University in 2003. Then he continued for the PhD program. He has been a teaching assistant and a research assistant for the department of Systems Engineering and Operations Research at GMU since 2001. His research interests include stochastic modeling, probabilistic inference, Bayesian networks, and simulation.