2007

Path propagation : a probabilistic inference algorithm for large and complex Bayesian networks

Liu He

University of Windsor

Path Propagation: A Probabilistic Inference Algorithm for Large and Complex Bayesian Networks

by

Liu He

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2007

Your file Votre référence
ISBN: 978-0-494-34950-2
Our file Notre référence
ISBN: 978-0-494-34950-2

ABSTRACT

Bayesian networks are widely used for knowledge representation and uncertain reasoning. One of the most important services which Bayesian networks provide is (probabilistic) inference. Effective inference algorithms have been developed for probabilistic inference in Bayesian networks for many years. However, the effectiveness of the inference algorithms depends on the sizes of Bayesian networks. As the sizes of Bayesian networks become larger and larger in real applications, the inference algorithms become less effective and sometimes are even unable to carry out inference.

In this thesis, a new inference algorithm specifically designed for large and complex Bayesian networks, called *path propagation*, is proposed. Path propagation takes full advantage of one of the most popular inference algorithms, i.e., global propagation. It improves over global propagation by carrying out inference only in certain paths in a junction tree that are relevant to queries. Compared with global propagation, path propagation takes less computational resources and can effectively improve the computational efficiency for inference in large and complex Bayesian networks.

DEDICATION

To my parents and all who love me and beloved

ACKNOWLEDGEMENTS

First of all, I really appreciate the great help of my supervisor, Professor Dan Wu, during my two years' master study in University of Windsor. Without his constant encouragement and valuable guidance, I could not make such success in my research field.

Secondly, I would also like to acknowledge Dr. Frank Jensen from Hugin Ltd. and Dr. Finn V. Jensen from Aalborg University. Their wisdom guidance helped me solve many difficult problems during my research.

Thirdly, I want to acknowledge my thesis committee members, Dr. Yunbi An, Dr. Angela C. Sodan, and Dr. Ziad Kobti for their unwavering help.

Finally, I am very grateful for my parents, Yiwen He and Guangyu Liu. During these two years' study, they gave me great confidence and solid support whenever I met any kind of problems. They always stand by my side and back me up no matter what happen to me. This is the most valuable asset in my life. Sincerely wish them have great happiness in their lives!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Over the past decade, Bayesian networks have been widely exploited in knowledge representation and reasoning in uncertainty management [27]. A Bayesian network is a directed acyclic graph augmented with conditional probability tables. It can be viewed as a graphical representation of a domain with uncertainty. Probabilistic inference simply means computing posterior probability distribution $P(X|Y=y)$ for variable(s) X of interest given the evidence that some other variable(s) Y is taking the value y. Computing $P(X|Y=y)$ is also called a query in the thesis. Probabilistic inference algorithms can be categorized into two kinds, exact inference and approximate inference. In this thesis, we focus on algorithms for exact inference. Henceforth, the term inference in this thesis refers to exact inference unless otherwise specified. Although the problem of probabilistic inference is NP-hard in general [17][18], many practically effective inference algorithms have been developed and implemented [5], for example, variable elimination [12], the Sum-Product algorithm [11], the arc reversal algorithm [15][16]. Among all inference algorithms that have so far been proposed, the most popular and successful algorithms are based on the idea in [27]. It converts a Bayesian network first into a secondary structure which is a junction tree, and then carries out inference on the junction tree. There are three architectures based on this idea, namely, the Lauritzen-Spiegelhalter (LS) architecture [14], the Hugin architecture [28][29], and the Shenoy-Shafer (SS) architecture [30][31]. Among these three different architectures, the Hugin architecture is not only technically superior but also commercially successful. The Hugin Tool [37][38], based on Hugin architecture by the Hugin Ltd., is popularly applied in various domains. It implements the renowned global propagation method which carries out inference on the junction tree.

However, for global propagation, it has several shortcomings. It only performs quite well on Bayesian networks within 1000 nodes. For large and complex Bayesian networks, the response speed of performing global propagation is quite slow. Also, it wastes lots of computational resources. In the thesis, path propagation, based on the Hugin architecture, is proposed. It carries out inference only in certain paths in a junction tree that are relevant to queries. It is based on the query imposed by users and it answers the query only. From this point of view, path propagation takes less computational resources than global propagation and can effectively improve the computational efficiency for inference in large and complex Bayesian networks.

The following explains the motivation of this thesis. Furthermore, the contributions of this thesis are highlighted and the structure of the remaining chapters is outlined.

1.1 MOTIVATION

Recently, researchers notice that the Bayesian network model has inherent deficiencies in its modeling capacity for large and complex domains [26]. Many extensions of the Bayesian network model have been proposed, such as the multiply sectioned Bayesian network (MSBN) model [21][22] and the object-oriented Bayesian network (OOBN) model [19][20]. The MSBN model and the OOBN model are not focusing on developing completely new methods for inference but focusing more on providing methodologies for modeling large and complex Bayesian networks. In fact, in order to conduct inference on these two models, one transforms them into Bayesian networks first, and then apply the regular inference algorithm on the transformed Bayesian network [2]. However, a single Bayesian network converted from either an OOBN or a MSBN could be very large and complex that the existing inference algorithms could not be effectively and efficiently applied. This presents challenge and opportunity to develop new inference algorithms that are specifically tailored to large and complex Bayesian networks.

Our experiments have shown that global propagation performs quite well on many Bayesian networks in practice within 1000 nodes, however, when dealing with large and complex Bayesian networks with more than 1000 nodes, problems occur and global propagation crashes. There are three possible reasons. First of all, performing global propagation may not be possible due to large and complex Bayesian networks. One of the experiments conducted on a Bayesian network with about 1300 nodes returns an out of memory error message on a P4 machine with 512 MB memory and normal load [25]. Secondly, the response speed of performing global propagation is slow on large and complex Bayesian networks, which means that the response time will be much longer. Thirdly, applying global propagation on large and complex Bayesian networks wastes many computation resources. For global propagation, it carries out inference over the whole junction tree and then the probability distribution of any variable of interest in a Bayesian network can be known thereinafter [44]. However, it is unreasonable to presume that users interest in the probability distributions of all variables in many real applications. In practice, users often interest in the probability distributions of a few variables in a Bayesian network, so performing global propagation on the whole junction tree only for a few variables of interest will surely waste lots of computational resources from this point of view.

1.2 CONTRIBUTION

Path propagation can well solve the shortcomings presented in Section 1.1, such as the waste of computational resources, the slow response speed for inference in large and complex Bayesian networks.

The first contribution for path propagation is that performing path propagation takes much shorter time than performing global propagation. Because the former only involves certain paths in a junction tree while the latter carries out inference in the whole junction tree. Obviously, computations caused by the former will be fewer than computations caused by the latter, which indicates that performing path propagation is more computational efficient than performing global propagation specifically in large and complex Bayesian networks.

Secondly, performing path propagation on large and complex Bayesian networks takes less computational resources than global propagation. After performing global propagation on the whole junction tree, the probability distribution of any variable of interest in a Bayesian network can be known thereinafter. As a matter of fact, users often interest in the probability distributions of a few variables in a Bayesian network, so performing global propagation on the whole junction tree only for a few variables of interest will surely waste lots of computational resources. However, for path propagation, if users interest a variable, a path will be determined in a junction tree and inference is carried out only on this path. So path propagation is based on a query which is imposed by users and it only answers the query. It takes less computational resources than global propagation.

1.3 ORGANIZATION

The rest of this thesis is organized as follows:

- Chapter 2 reviews the background knowledge of Bayesian networks. At the end of this chapter, the LS architecture, the Hugin architecture, and the SS architecture are introduced.
- Chapter 3 proposes path propagation with two fundamental theorems and scenarios. Then the prototype of path propagation is presented. At last, the analysis of these two scenarios is provided.
- Chapter 4 describes the general implementation information and provides the experimental results. These experimental results are compared in this chapter and evaluations are obtained.
- Chapter 5 concludes the thesis and discusses the directions of future work.

CHAPTER II

BACKGROUND KNOWLEDGE

This chapter briefly reviews the fundamentals of probability theory and Bayesian networks, such as the definition of Bayesian networks, the notations related to Bayesian networks, and the general inference procedure in Bayesian networks. At the end of this chapter, three existing architectures for inference in Bayesian networks are discussed.

2.1 PROBABILITY THEORY

Probability theory is used as the mathematical research of phenomena which is uncertain. More specifically, when an experiment which is conducted under the same circumstances produces different outcomes, *probability* is used to model these outcomes. In probability theory, the outcome of an experiment may not certain and all of its possible outcomes are expectable in advance. Therefore, the set of all these possible outcomes of an experiment is called the *sample space* of the experiment and is often denoted as S. All these possible outcomes of S should be *mutually exclusive* and *exhaustive*. The term *mutually exclusive* means that all these possible outcomes of S can not overlap, while the term *exhaustive* means that S must include all possible outcomes. For example, from the coin-tossing experiment, S is composed of two outcomes which are "heads" and "tails", written as S={heads, tails}. In the language of probability, *events* are use to represent certain subsets of S. In that case, consider a set of events X, a probability is a numerically valued function that assigns a real number denoted as $P(X)$ to X and $P(X)$ should be between 0 and 1. Therefore, $P(X)$ is called the probability of X. If there is another set of events Y, the probability of X given the occurrence of Y is called the *conditional probability* of X given Y, written as $P(X|Y)$. Suppose an event happens whenever X and Y happen at the same time, it is called the intersection of X and Y, denoted as XY or $X \cap Y$. Therefore, the conditional probability of X given Y is defined as $P(X \mid Y) = \frac{P(XY)}{P(Y)}$, provided $P(Y)>0$

[45][46]. In probability theory, probabilities are usually very sensitive given some conditioning information. However, sometimes a probability remains unchanged when given the conditioning information. To make the discussion more concrete and manageable, given two sets of events X and Y, if the probability of X remains unchanged with knowing the occurrence of Y, written as $P(X|Y)=P(X)$, X and Y are said to be *independent* to each other. Suppose there is another set of events Z, if X and Y are said to be *conditionally independent* given Z, then the probability of X given Z remains unchanged with knowing the occurrence of Y, written as $P(X|Y, Z)=P(X|Z)$. In this equation, the comma "," represents logic conjunction "and" which is denoted as the symbol "$\cup$".

In probability theory, most experiments that we encounter produce outcomes that can be interpreted in terms of real numbers, so these numerical outcomes whose values can vary from experiment to experiment are so called *random variables*, that is, a random variable is a real-valued function on S [46]. In general, random variables may be discrete and continuous. However, we only consider discrete random variables here. Henceforth, the term random variables in this thesis refer to discrete random variables unless otherwise specified. In the thesis, random variables are denoted with boldface italic uppercase letters, e.g., $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$, and their actual values are represented as bold lowercase letters, e.g., $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$. The values of a random variable should be *mutually exclusive* and *collectively exhaustive*. For example, there is a random variable $\boldsymbol{X}$ only containing two values $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$. If $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are mutually exclusive, they can not both happen simultaneously. If $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are collectively exhaustive of $\boldsymbol{X}$, they should encompass the whole range of possible values of $\boldsymbol{X}$. Therefore, a *probability distribution* of a random variable, also called a probability distribution function of a random variable, maps each possible values of a random variable into real numbers and all these numbers should be between 0 and 1. Furthermore, all these numbers must add to 1. Two distributions in probability theory, called *conditional probability distribution (CPD)* and *joint probability distribution (JPD)*, are two of the most important notations in Bayesian network. They will be discussed in Section 2.2.2.

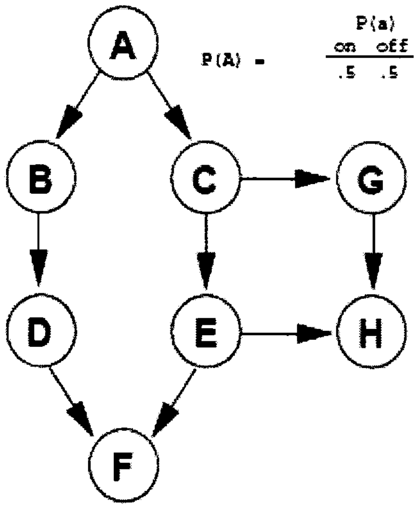
2.2 BAYESIAN NETWORKS

This section briefly reviews the background knowledge of Bayesian networks and discusses inference in Bayesian networks.

2.2.1 A Brief Introduction of Bayesian Networks

A Bayesian network is an effective and powerful framework widely used in knowledge representation and reasoning under uncertainty [8]. It is a kind of graphical model. Graphical models include *undirected graphical models* and *directed graphical models*, where nodes are used to represent random variables while arcs are used to represent the probabilistic dependence between a node and its parents. Undirected graphical models, also named *Markov Random Fields*, have a straightforward definition of independence: two (sets of) nodes called X and Y are conditionally independent when given a third set called Z, if all paths between the nodes in X and Y are separated by a node in Z [3][4]. For directed graphical modes, also called *Bayesian networks*, they can not have directed cycles. Therefore, the definition of independence for directed graphical models has a more complicated concept than undirected graphical models because of adding the directionality of the arcs. More details about independence in Bayesian networks are discussed in Section 2.2.2.

A Bayesian network contains two components called the graphical component and the numerical component. The graphical component is a *directed acyclic graph* (DAG) used to indicate direct influences among variables. Nodes are used to indicate random variables and directed arrows or links are used to connect two nodes. In this thesis, we will use the terms "nodes" and "variables" interchangeably hereafter. The numerical component is a set of *conditional probability tables* (CPTs). Each node in a Bayesian network is attached with the conditional probability of the node given its parents. Figure 1 illustrates a small example of a Bayesian network [5]. The left side of Figure 1 illustrates the graphical component of the Bayesian network which is a DAG, while the right side of Figure 1 illustrates a set of CPTs of the Bayesian network.



P(A) =

P(a) on	P(a) off
.5	.5

P(B|A) =

a	P(b|a) on	off
on	.5	.5
off	.4	.6

P(C|A) =

a	P(c|a) on	off
on	.7	.3
off	.2	.8

P(D|B) =

a	P(d|b) on	off
on	.9	.1
off	.5	.5

P(E|C) =

a	P(e|c) on	off
on	.3	.7
off	.6	.4

P(F|DE) =

d	e	P(f|de) on	off
on	on	.01	.99
on	off	.01	.99
off	on	.01	.99
off	off	.99	.01

P(G|C) =

c	P(g|c) on	off
on	.8	.2
off	.1	.9

P(H|EG) =

g	h	P(h|eg) on	off
on	on	.05	.95
on	off	.95	.05
off	on	.95	.05
off	off	.95	.05

Figure 1: An example of a Bayesian network.

2.2.2 Basic Notations of Bayesian Networks

In Bayesian networks, possible outcomes of a *variable* can be also called *states*. These states for a random variable are mutually exclusive and collectively exhaustive. For example, consider a variable X associated with three values x_1, x_2 and x_3, if a value x_2 is assigned to X, this operation is called that x_2 is an *instantiation* of X and the value x_2 to X is called *evidence* which means an observed instantiation of some random variables. Evidence is denoted as e in the thesis. In the thesis, sets of variables are represented by italic uppercase letters, e.g., X, Y, Z, and their instantiations are denoted by italic lowercase letters, e.g. x, y, z. Thus, more precisely, a set of variables Y are instantiated through assigning a value to each variable in Y and this assignment is denoted by y. So y is called an instantiation of Y and y to Y is called evidence. Consider two pieces of evidence e_1 and e_2. If there exists a variable X which is included in both pieces of evidence and X is taking same value in e_1 and e_2, then these two pieces of evidence are called *compatible*, otherwise, called *contradicting* [25].

A *potential* is defined as a function over Y that maps each instantiation y into a nonnegative real number. It can be viewed as matrices and implemented as tables. The symbol "ϕ" is used to represent potential here. In Bayesian networks, there are two basic and important operations on potentials, called *multiplication* and *marginalization*.

Suppose there are two sets of variables called X and Y. Their corresponding potentials are $\phi(X)$ and $\phi(Y)$. The *multiplication* of $\phi(X)$ and $\phi(Y)$ is a potential $\phi(Z)$ where $Z=X \cup Y$, defined as $\phi(Z) = \phi(X) \cdot \phi(Y)$. If $X \subseteq Y$, the *marginalization* of $\phi(Y)$ onto X is a potential $\phi(X)$, defined as $\phi(X) = \sum_{Y-X} \phi(Y)$, where the operator "–" denotes set difference.

Given two sets of variables X and Y, *the conditional probability distribution (CPD)* of X given an evidence of $Y=y$ is the probability distribution of X conditioned on the assumed value y of Y. *Prior probability* means the probability of a variable $\boldsymbol{X}$ without given any evidence. It can be written as $P(\boldsymbol{X})$. *Posterior probability* means the probability of a variable $\boldsymbol{X}$ given evidence e. It can be written as $P(\boldsymbol{X}|e)$. *Joint probability distribution (JPD)* maps each of the combinations of all possible values of all variables to nonnegative real number and all these numbers should be between 0 and 1. Furthermore, the sum of these numbers must add to 1. In Bayesian networks, JPD can be calculated as:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid parents(X_i))$$

Where $\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots, \boldsymbol{X_n}$ represent all nodes as well as all variables in Bayesian networks and *parents*(X_i) means the parents of $\boldsymbol{X_i}$.

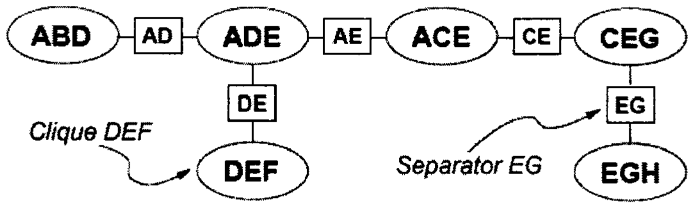
In Bayesian networks, a very important concept is *independence assertions* which are encoded in the DAG. The construction of Bayesian networks is based on independence assertions. If X and Y are conditionally independent given Z, for all combinations of values x, y, and z, an *independence assertion* is a statement of the form which holds $P(x|z)=P(x|y, z)$, denoted as $I(X, Z, Y)$ where Z is not empty. More concretely, even though z is assigned to Z, the probability of x given z is not affected by knowing y. If Z is empty, then X and Y are said to be independent to each other, denoted as $I(X, \varnothing, Y)$ where the symbol "$\varnothing$" denotes empty set. In Bayesian networks, independence assertions indicate every variable is conditionally independent of its non-descendants given its parents. Thus, through a graphical criterion called *d-separation* [6], a DAG can encode all these independence assertions. Independence assertions are very important in Bayesian networks because the renowned *Probability Propagation in Trees of Clusters* (PPTC) method, introduced in the next section, uses them to reduce the complexity for inference in Bayesian networks [5].

2.3 INFERENCE IN BAYESIAN NETWORKS

Probabilistic inference simply means computing posterior probability distribution $P(X|Y=y)$ for variable(s) X of interest given the evidence that some other variable(s) Y is taking the value y. Generally speaking, there are two kinds of inference. One is about inference with no evidence observed (e.g. $P(X)$) and another is about inference with evidence observed (e.g. $P(X \mid e)$), where X denotes a set of variables and e denotes evidence. Both of them will be briefly introduced in this section. A very popular probabilistic inference method called the *Probability Propagation in Trees of Clusters* (PPTC) can be applied on these two kinds of inference. It contains synthesized approaches which are scattered throughout the literature and changes these approaches to algorithmic form [5]. PPTC includes two steps for inference in Bayesian networks. First of all, a Bayesian network is converted into a secondary structure which is a junction tree. Secondly, inference is carried out on the junction tree by performing some popular algorithms, such as the renowned method called global propagation. Based on this idea, the LS architecture, the Hugin architecture and the SS architecture have been proposed over the past decade. Among these three architectures, the Hugin architecture is not only technically superior but also commercially successful. The Hugin Tool, based on Hugin architecture by the Hugin Ltd., is popularly applied in various domains for inference in Bayesian networks and other tasks. Thus, we introduce these three architectures with a focus on the Hugin architecture in this chapter.

2.3.1 The Conversion of a Bayesian Network into a Junction Tree

A *junction tree* includes two components, the graphical component and the numerical component. For the graphical component, it is an *undirected tree*. Each node in the tree represents a *clique* containing a set of variables, while each edge in the tree represents an intersection of two adjacent cliques which is called a *separator*. Consider two adjacent cliques C_i and C_j. So their separator S_{ij} equals $C_i \cap C_j$. The size of a clique or separator means the product of all states of all variables contained in a clique or separator. Cliques in a junction tree should satisfy the *junction tree property* that all cliques on the path between any clique X and any clique Y have to contain the intersection of X and Y, e.g., $X \cap Y$ [5][9]. For the numerical part, each clique is associated with a potential. However, these potentials in a junction tree are not arbitrarily defined. They have to satisfy a constraint to make the junction tree *locally consistent*, which means for each clique X associated with $\phi(X)$ and its neighboring separator S associated with $\phi(S)$, this equation $\phi(S) = \sum_{X-S} \phi(X)$ should be satisfied. Also, these potentials have to satisfy another constraint that the joint probability distribution $P(U)$ can be denoted by these potentials through the equation $P(U) = \frac{\prod_i \phi(X_i)}{\prod_j \phi(S_j)}$, where U includes all random variables in a Bayesian network and $\phi(X_i)$ and $\phi(S_j)$ represents the clique and separator potentials respectively. If the potentials in a junction tree satisfy these two constraints, for each clique (or separator) X, it holds this equation: $\phi(X) = P(X)$ [29]. Hence, if a user interests in any variable in the junction tree and a clique (or separator) is identified that it contains this variable, the probability distribution of this variable can be obtained from marginalization of the clique (or separator) potential onto this variable. Figure 2 illustrates an example of a junction tree. The top half of Figure 2 illustrates the undirected tree while the bottom half of Figure 2 gives two examples of potentials associated with a clique and a separator in the undirected tree.



ϕ_{ABD} =

a	b	d	$\phi_{ABD}(abd)$
on	on	on	.225
on	on	off	.025
on	off	on	.125
on	off	off	.125
off	on	on	.180
off	on	off	.020
off	off	on	.150
off	off	off	.150

ϕ_{AD} =

a	d	$\phi_{AD}(ad)$
on	on	.35
on	off	.15
off	on	.33
off	off	.17

etc.

Figure 2: An example of a junction tree.

Generally speaking, there are four steps to convert a Bayesian network into a junction tree, called *moralization*, *triangulation*, *identification of maximal cliques,* and *building an optimal junction tree*. In order to make these four steps clearly and concretely, we take Figure 1 as an example to show how to convert Figure 1 into a junction tree through the four steps.

Step 1: Moralization

First of all, delete the directions of all arcs in Figure 1 to convert it into an undirected graph. The undirected graph is illustrated in Figure3.

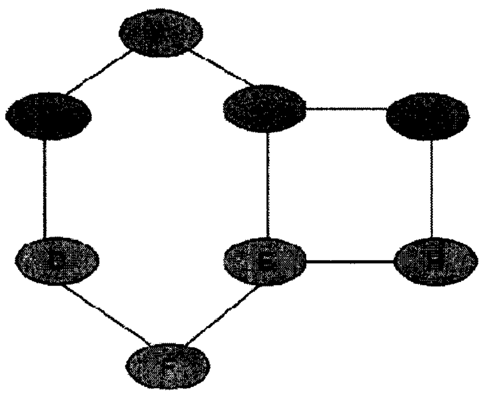


Figure 3: The undirected graph after deleting the directions of the arcs in Figure 1.

Secondly, for each node in Figure 3, identify its parents through Figure 1, and then connect each pair of its parents by adding an undirected arc. After that, the undirected graph becomes the moralized graph. Figure 4 is the moralized graph of Figure 3. Dashed lines in Figure 4 represent the undirected arcs added to Figure 3.

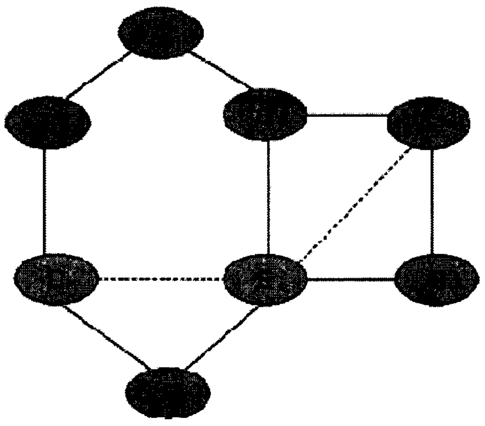


Figure 4: The moralized graph of Figure 3.

Step 2: Triangulation

If the moralized graph is triangulated, then every cycle of length four or greater in this graph should contain an undirected edge which connects two nonadjacent nodes in the cycle [9]. If not, undirected arcs are added to make the graph satisfy the criteria. The *elimination ordering* [5] algorithm is used to selectively add undirected arcs to make the moralized graph triangulated, so cliques can be identified from the triangulated graph. Generally speaking, many existing algorithms can be used to triangulate a moralized graph. Elimination ordering algorithm is one of the optimal triangulation methods which minimize the sum of the sizes of the cliques obtained from the triangulated graph. Figure 5 illustrates the triangulated graph of Figure 4. Red lines in Figure 5 represent the undirected arcs added to Figure4.

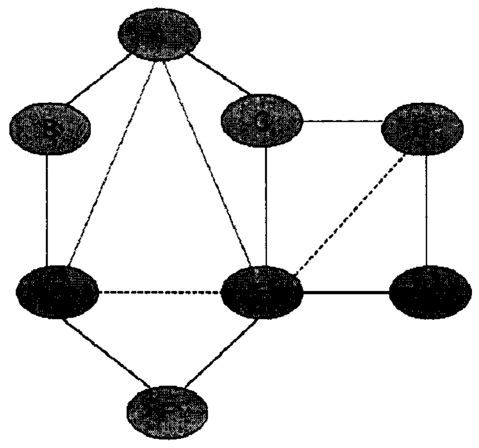


Figure 5: The triangulated graph of Figure 4.

Step 3: Identification of Maximal Cliques

From the triangulated graph, *complete* and *maximal* cliques have to be identified. Generally speaking, a clique is a sub-graph of an undirected graph. Since the triangulated graph is an undirected graph, cliques can be identified from the triangulated graph. In order to make cliques *complete* and *maximal*, make sure that each pair of different nodes in a clique must be linked by an edge and the clique is not enclosed in a larger complete sub-graph. Therefore, six complete and maximal cliques are obtained from Figure 5, clique EGH, clique CEG, clique DEF, clique ACE, clique ABD, and clique ADE.

Step 4: Building an Optimal Junction Tree

After obtaining these six cliques in step 3, a junction tree can be built. In order to satisfy the junction tree property, different ways can be used to link these six cliques to produce different junction trees. In General, given a set of k cliques, first create an empty separator set S. For each distinct two cliques X and Y, create a separator S_{XY} that S_{XY} equals the intersection of X and Y. Then S_{XY} is inserted into S. At last, separators in S are iteratively inserted between pairs of cliques until all k cliques are connected by k-1 separators. A general algorithm called *Building an Optimal Junction Tree* [5] can select k-1 separators in S to obtain an optimal junction tree. An optimal junction tree can minimize the computational time needed for probabilistic inference. Figure 6 illustrates an optimal junction tree constructed by the six cliques obtained from Figure 5.

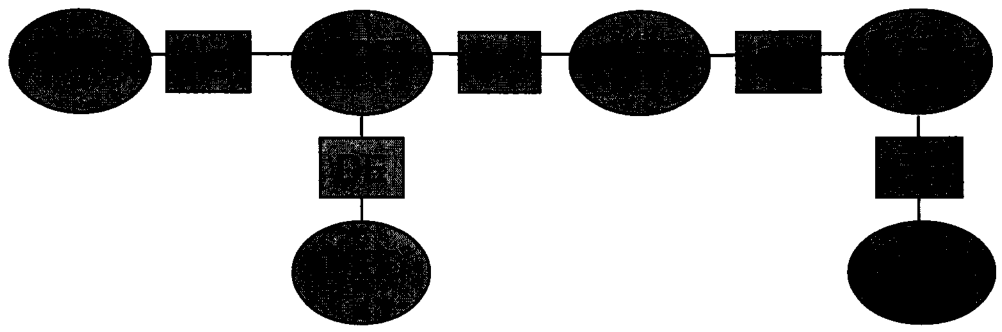


Figure 6: An optimal junction tree constructed by the six cliques obtained from Figure 5.

After obtaining the optimal junction tree, the potential of each clique in the junction tree is initialized to 1 in the first place. Then for each clique in the junction tree, a set of CPTs are assigned to obtain its potential. The way to assign the CPTs into cliques is described as follows. For each variable and its parents in the original Bayesian network, if they are all contained in a clique in the junction tree, then this variable's CPT is assigned to the clique. After assigning every variable's CPT into a corresponding clique, for each clique, we multiply these CPTs to obtain the potential of the clique. However, these potentials can not satisfy the constraint that the junction tree must be locally consistent. Therefore, some algorithm (e.g., global propagation) needs to be applied on these potentials to make the junction tree locally consistent. More details will be discussed in the next section.

2.3.2 Global Propagation on a Junction Tree

For PPTC, an algorithm called *global propagation* performs a series of ordered *message* *passes* on the junction tree to make it locally consistent. A message pass which is also called a local manipulation happens between a clique X and a neighboring clique Y though their separator S. Thus, after completing global propagation, the junction tree is locally consistent. So probabilities can be calculated on the locally consistent junction tree.

First of all, a massage can be calculated between two adjacent cliques through their separator. Consider two adjacent cliques C_i and C_j with their separator $S_{ij} = C_i \cap C_j$. Their associate potentials are $\phi(C_i)$, $\phi(C_j)$, $\phi(S_{ij})$. The message pass from clique C_i to clique C_j through their separator S_{ij} can be calculated in two steps called *projection* and *absorption*.

Projection:

Save the old potential table $\phi(S_{ij})$ to $\phi^{old}(S_{ij})$ first, denoted as $\phi^{old}(S_{ij}) \leftarrow \phi(S_{ij})$. Then calculate a new potential table to S_{ij} from clique C_i and store it to $\phi(S_{ij})$, defined as $\phi(S_{ij}) \leftarrow \sum_{C_i - S_{ij}} \phi(C_i)$.

Absorption:

Assign a new potential table to clique C_j, using both old and new potential tables of S_{ij}, defined as $\phi(C_j) \leftarrow \phi(C_j) \cdot \frac{\phi(S_{ij})}{\phi^{old}(S_{ij})}$.

Combining projection and absorption, the following formula is generated for calculating a message pass between two adjacent cliques.

$$\phi(C_j) = \phi(C_j) \cdot \frac{\sum_{C_i - S_{ij}} \phi(C_i)}{\phi(S_{ij})}$$

Secondly, calculated messages can be passed by global propagation. It contains three steps. Choose an arbitrary clique X in a junction tree at first. Then unmark all cliques and call an algorithm named *Collect-Evidence(X)* [10]. At last, unmark all cliques and call an algorithm named *Distribute-Evidence(X)* [10].

For *Collect-Evidence(X)*, *X* is marked first and then *Collect-Evidence* is called recursively on *X*'s unmarked neighboring cliques, if any. At last, a message is passed from *X* to the clique invoked *Collect-Evidence(X)*. For *Distribute-Evidence(X)*, *X* is marked first and then a message is passed from *X* to each of its unmarked neighboring cliques, if any. At last, *Distribute-Evidence* is called recursively on *X*'s unmarked neighboring cliques, if any. Thus, global propagation passes messages within the whole junction tree and calculates messages between every two adjacent cliques. Figure 7 illustrates how to pass messages using global propagation in Figure 6 [5]. Arrows in Figure 7 represent the procedure of message passes in Figure 6. The root clique in Figure 7 is clique ACE. Solid arrows represent *Collect-Evidence* while dashed arrows represent *Distribute-Evidence.* Numbers denote the message passing sequence.

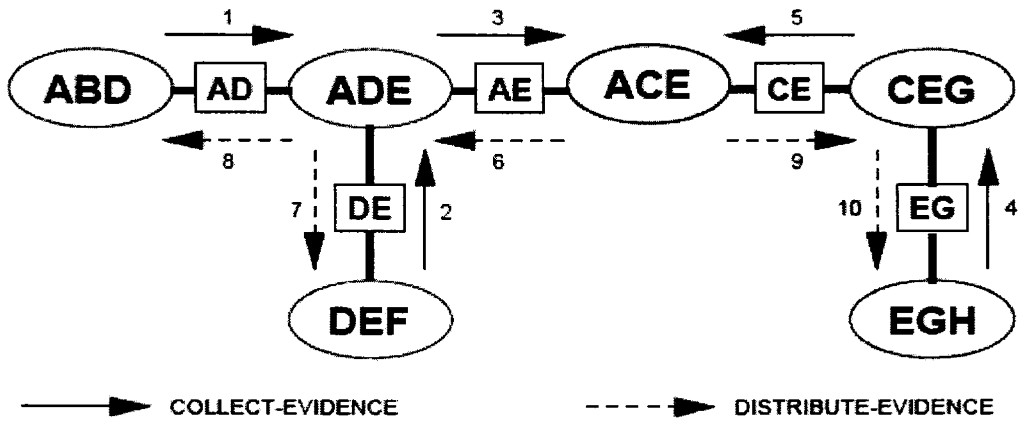


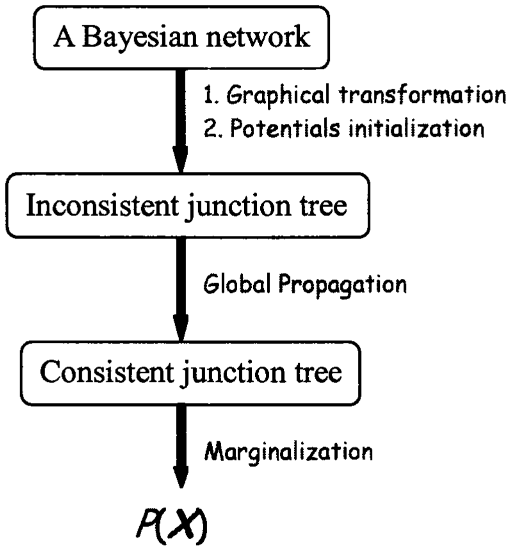
Figure 7: How to pass messages using global propagation in Figure 6.

After performing global propagation on the junction tree, it becomes locally consistent. $P(C_i)$ and $P(S_j)$ can be obtained for every clique and separator potential in the junction tree, which means these cliques and separator are marginalized. Hence, they can be called marginals. Therefore, probability distributions can be calculated through these marginals. As we have mentioned before, there are two kinds of inference. One is about inference with no evidence observed (e.g. $P(X)$) and another is about inference with evidence observed (e.g. $P(X \mid e)$). More details about these two inference procedures are introduced in the following two sections.

2.3.3 Inference with No Evidence Observed

Suppose a variable X is interested in a Bayesian network. If there is no evidence observed, four steps are performed to obtain the prior probability distribution of X of interest. They are listed as follows and Figure 8 illustrates the process of inference with no evidence observed.

i. Convert a Bayesian network into an optimal junction tree, described in Section 2.3.1.
ii. Initialize all potentials in the junction tree, introduced in Section 2.3.1.
iii. Perform global propagation on the whole junction tree, discussed in Section 2.3.2
iv. Find a marginalized separator that contains X, say $P(S_i)$. Then perform marginalization (introduced in Section 2.2.2) on this separator to return the prior probability distribution of X of interest, say $P(X)$. If no separator in the junction tree contains X, then find a marginalized clique that contains X, say $P(C_i)$ and apply marginalization on this clique to return $P(X)$.


A Bayesian network
1. Graphical transformation
2. Potentials initialization
Inconsistent junction tree
Global Propagation
Consistent junction tree
Marginalization
P(X)
Figure 8: Inference with no evidence observed.

2.3.4 Inference with Evidence Observed

Suppose the posterior probability distribution of a variable X given evidence e is interested in a Bayesian network. In order to return the posterior probability distribution of X given e, a new notation called the *likelihood* is introduced. The likelihood of a variable is a potential that encodes each value of this variable into a real number, denoted as Λ. The real number can only be 0 or 1. For example, consider a variable Y containing three values y_1, y_2, and y_3, if y_2 is observed to Y as evidence, then $\Lambda(Y)=\begin{cases}0, Y=y_1\\ 1, Y=y_2\\ 0, Y=y_3\end{cases}$.

However, if no values of Y is observed which means Y is not the evidence, then $\Lambda(Y)=\begin{cases}1, Y=y_1\\ 1, Y=y_2\\ 1, Y=y_3\end{cases}$. Figure 9 illustrates the example of the likelihood of all variables in Figure 1 given the evidence including $\boldsymbol{B}$=off, $\boldsymbol{E}$=on, and $\boldsymbol{H}$=off.

$$\Lambda \Rightarrow \left\{\begin{array}{ll}
\Lambda(A) \Rightarrow & \begin{cases}1, A=on\\ 1, A=off\end{cases}\\
\Lambda(\boldsymbol{B}) \Rightarrow & \begin{cases}0, \boldsymbol{B}=on\\ 1, \boldsymbol{B}=off\end{cases}\\
\Lambda(C) \Rightarrow & \begin{cases}1, C=on\\ 1, C=off\end{cases}\\
\Lambda(D) \Rightarrow & \begin{cases}1, D=on\\ 1, D=off\end{cases}\\
\Lambda(\boldsymbol{E}) \Rightarrow & \begin{cases}1, \boldsymbol{E}=on\\ 0, \boldsymbol{E}=off\end{cases}\\
\Lambda(F) \Rightarrow & \begin{cases}1, F=on\\ 1, F=off\end{cases}\\
\Lambda(G) \Rightarrow & \begin{cases}1, G=on\\ 1, G=off\end{cases}\\
\Lambda(\boldsymbol{H}) \Rightarrow & \begin{cases}0, \boldsymbol{H}=on\\ 1, \boldsymbol{H}=off\end{cases}
\end{array}\right.$$

Figure 9: The example of the likelihood of all variables in Figure 1 given the evidence including $\boldsymbol{B}$=off, $\boldsymbol{E}$=on, and $\boldsymbol{H}$=off.

In order to return the posterior probability distribution of X given evidence e, there are six steps described as follows and Figure 10 illustrates the process of inference with evidence observed:

i. Convert a Bayesian network into an optimal junction tree, described in Section 2.3.1.

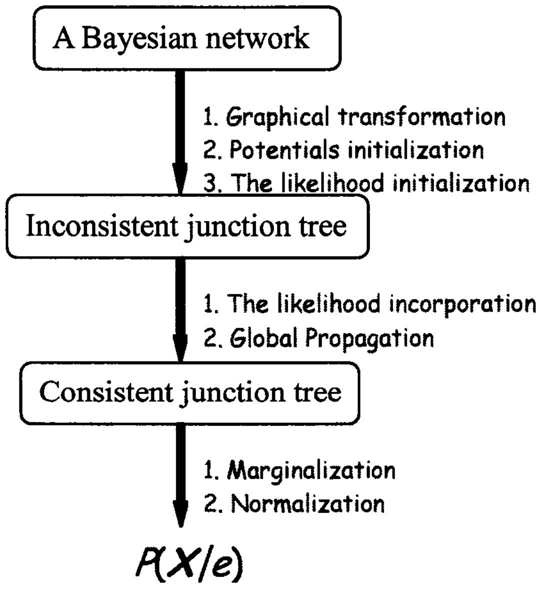
ii. Initialize all potentials in the junction tree, introduced in Section 2.3.1. Also, initialize the likelihood of each variable according to evidence *e*.

iii. Incorporate the likelihood into the junction tree, that is, for each variable, if it is included in *e*, its likelihood will be multiplied to the potentials that contain this variable.

iv. Perform global propagation on the whole junction tree, discussed in Section 2.3.2

v. Find a marginalized separator that contains X, say $P(S_i, e)$. Then perform marginalization (introduced in Section 2.2.2) on this separator to return the prior probability distribution of X of interest, say $P(X, e)$. If no separator in the junction tree contains X, then find a marginalized clique that contains X, say $P(C_i, e)$ and apply marginalization on this clique to return $P(X, e)$.

vi. Apply normalization (introduced in Section 2.2.2) on $P(X, e)$ to return the posterior probability distribution of X of interest given e followed by the formula: $P(X \mid e) = \frac{P(X,e)}{P(e)} = \frac{P(X,e)}{\sum_X P(X,e)}$.



A Bayesian network
1. Graphical transformation
2. Potentials initialization
3. The likelihood initialization
Inconsistent junction tree
1. The likelihood incorporation
2. Global Propagation
Consistent junction tree
1. Marginalization
2. Normalization
P(X/e)

Figure 10: Inference with evidence observed.

2.4 THE HUGIN ARCHITECTURE

The Hugin architecture introduces a separator which is the intersection of two adjacent cliques. It saves lots of computations by using separators [23]. The procedure for inference in the Hugin architecture includes two passes called *Inward Pass* and *Outward Pass*. It can be described as follows.

i. In order to save more computations, a clique with *maximal size* is picked as a root in the junction tree.
ii. After that, direct all arcs in the junction tree toward the root clique.
iii. For each node, send a message to its inward neighbor in the root clique's direction, starting from the cliques which are furthest from the root clique. This is called Inward Pass.
iv. For each node, send a message to its outward neighbors away from the root clique's direction, starting from the root clique itself. This is called Outward Pass.

For Inward Pass, it is followed by six rules cited from [1].

Inward Pass:

Rule 1: When each non-root node has received messages from all its other neighbors, then it will send its message to a given neighbor.

Rule 2: When the root has received messages from all its neighbors, it will send messages to them.

Rule 3: A node computes the message by marginalizing its current potential to its intersection with this neighbor when it is ready to send its message to a particular neighbor, and then it sends this message to the separator between it and the neighbor.

Rule 4: A separator will divide the message by its *current potential* when it receives a message containing the *new potential* from one of its two nodes, send the quotient (*new potential)/(current potential)* on to the other node, and then replace *current potential* with *new potential.*

Rule 5: A node replaces its current potential with the product of the potential and the message when it receives a message.

Rule 6: When the root has received a message from all its outward neighbours, Inward Pass is end.

For Outward Pass, it is followed by four rules cited from [1].

Outward Pass:

Rule 1: When each node receives a message from its inward neighbor, then it can send its messages to its outward neighbors. However, the root can send a message to its outward neighbors immediately if it does not have inward neighbors.

Rule 2: A node computes the message by marginalizing its current potential to its intersection with the outward neighbor when it is ready to send a message to its outward neighbor. It sends this message to its outward neighbor. Moreover, this new message will replace the old potential as the new potential in its separator.

Rule 3: A node replaces its current potential with the product of that potential and the message when it receives a message from its inward neighbor.

Rule 4: When all leaves have received messages from their inward neighbors, Outward Pass is end.

At the end of Inward Pass, the root clique is marginalized, while at the end of Outward Pass, every clique and separator in the junction tree are marginalized, e.g., $P(C_i)$ and $P(S_i)$. Thus, the probability distribution for a variable of interest with no evidence observed can be calculated from a smallest size separator that contains the variable through marginalization. Also, the posterior probability distribution for a variable of interest with evidence observed can be also calculated from a smallest size separator that contains the variable through marginalization and normalization. If there is no separator that contains the variable of interest, it can be computed from a smallest size clique that contains the variable of interest.

2.5 THE LS ARCHITECTURE AND SS ARCHITECTURE

The LS architecture is basically simpler than the Hugin architecture. The difference between these two architectures is that the LS architecture does not contain separators.

Figure 11 illustrates an example of the LS architecture based on Figure 1. In Figure 11, there are only six cliques and no separators. All computations with potentials are only calculated within cliques.

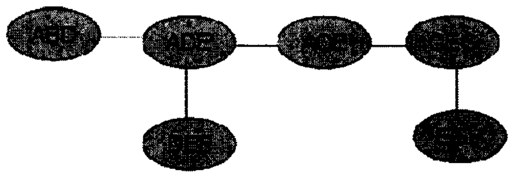


Figure 11: An example of the LS architecture based on Figure 1.

The procedure for inference in the LS architecture is almost the same as the Hugin architecture. It also includes two passes called *Inward Pass* and *Outward Pass*. Since the LS architecture does not contain separators, for Inward Pass of the LS architecture, a node calculates the message through marginalizing its current potential to its intersection with its inward neighbor when it is ready to send a message to the neighbor [1]. After that, this node sends this calculated message to the neighbor. Meanwhile, it divides its own current potential by this calculated message. At the end of Inward Pass, the root is marginalized, while each clique is marginalized at the end of Outward Pass, e.g., $P(C_i)$. Thus, the probability distribution for a variable of interest with no evidence observed can be calculated from a smallest size clique that contains the variable through marginalization. Also, the posterior probability distribution for a variable of interest with evidence observed can be also calculated from a smallest size clique that contains the variable through marginalization and normalization.

Unlike the LS and Hugin architecture, the number of neighbours of each node in the SS architecture is the focused point for this architecture. It uses a new data structure called a binary junction tree used for efficiently calculating multiple marginals. A binary junction tree is a junction tree that no node has more than three neighbours and all combinations are exploited in a binary junction tree on a binary basis [24]. In that case, the SS architecture caches computation to decrease the computation cost caused by combination and marginalization.

The procedure for inference in SS architecture also includes two passes called *Inward Pass* and *Outward Pass*. The procedure is cited from [1]:

i. In [35][36], a binary junction tree can be constructed by fusion algorithm [32][33]. It contains all singleton subsets.
ii. Associate each potential with one of the subsets in the binary junction tree according to its domain.
iii. Computing messages included both Inward Pass and Outward Pass: each node that needs to calculate the marginal requires a message from each of its neighbors in the tree.
iv. Computing marginals: if a node that needs to calculate the marginal has required and received messages from all its neighbors, then it can calculate the required marginal.

From the procedure of inference in SS architecture, it has no division operations. Furthermore, the original potentials remain unchanged in the SS architecture during the propagation procedure and the probability distribution of a variable of interest is calculated from the corresponding singleton variable node of the binary junction tree [34].

CHAPTER III

THEORETICAL ANALYSIS OF PATH PROPAGATION

For PPTC, a Bayesian network is converted into a junction tree and then inference is carried out on the whole junction tree by performing the renowned global propagation. In the thesis, path propagation is proposed for inference instead of global propagation.

3.1 SHORTCOMINGS OF GLOBAL PROPAGATION

The first shortcoming of global propagation is that global propagation can only be well performed in small or mid sized Bayesian networks within 1000 nodes in practice. According to our experiments on different Bayesian networks conducted by the Hugin Tool, inference in a large Bayesian network (the *Munin* network) with about 1300 nodes returns an out of memory error message on a P4 machine with 512 MB memory and normal load [25]. Since the effectiveness and efficiency of global propagation are strongly related to the size of a Bayesian network in reality, it is foresee that inference using global propagation takes much longer time on larger Bayesian networks. According to our experiments conducted by the Hugin Tool, the larger the size of a Bayesian network, the longer it will take to do the inference using global propagation.

The second shortcoming is that performing global propagation wastes many computation resources. According to the previous review of global propagation, the probability distribution (or posterior probability distribution when evidence is observed) for every variable of interest in a Bayesian network is readily available after a full scale global propagation. However, it is unreasonable to presume that users interest in the probability distribution (or posterior probability distribution when evidence is observed) for each variable in many real applications. To make it clear and concrete, take the *Munin2*

Bayesian network as an example. In *Munin2*, clinical tests are considered as the evidence and diseases are considered as random variables. Given these clinical tests, a doctor is able to make a diagnosis on the probabilities of a few interested variables (diseases). Since *Munin2* has 1003 variables (diseases), after a full scale global propagation on this network given the evidence (some clinical tests), the posterior probability distributions for each of the 1003 variables can be known. It is not a reasonable assumption that every variable in the network is interested whenever new evidence is observed. In practice, users often interest in the posterior probability distributions of a few variables in a Bayesian network, so many calculations caused by a full scale global propagation are totally wasted. Hence, it is not an economic way for inference especially in large and complex Bayesian networks.

Furthermore, recent researchers' trend is to extend Bayesian networks into large and complex Bayesian networks [25]. Many extensions of the Bayesian network model have been proposed, such as the multiply sectioned Bayesian network (MSBN) model and the object-oriented Bayesian network (OOBN) model. The MSBN model and the OOBN model are not focusing on developing completely new methods for inference but focusing more on providing methodologies for modeling large and complex Bayesian networks. In order to conduct inference on these two models, it was suggested to transform them into Bayesian networks first and then apply inference on the transformed Bayesian network [2]. However, since OOBN and MSBN models are originally exploited to model large and complex Bayesian networks, a single Bayesian network converted from either an OOBN or a MSBN model could be much larger and more complex than any existing Bayesian network. Hence, the existing inference algorithms could not be effectively and efficiently applied on them. Challenge and opportunity are presented to develop new inference algorithms which are specifically tailored to large and complex Bayesian networks.

In the thesis, path propagation is proposed to address the shortcomings of global propagation. It carries out inference only in certain paths in a junction tree that are relevant to queries. It is based on a query imposed by users and answers the query only.

From this point of view, it takes less computational resources than global propagation and improves the computational efficiency for inference in large and complex Bayesian networks. More details are introduced in the next three sections.

3.2 PATH PROPAGATION

Path propagation is developed specifically for inference in large and complex Bayesian networks. It inherits all novel features of global propagation. It is based on a query imposed by users and it only answers the query, so it takes less computational resources than global propagation. Furthermore, it only involves a path in a junction tree and carries calculations only in the path. Hence, it takes less time to answer queries than global propagation. The proposed propagation method is based on the *assumption* that a full scale global propagation is performed once on a junction tree with no evidence observed and thus the marginal for each clique or separator in the junction tree is obtained thereinafter. This assumption is justified by how to inference a Bayesian network in real applications. Because in practice, users first run inference with no evidence observed on the whole junction tree once to check the prior probability distribution for any variable of interest in a Bayesian network, say $P(X_i)$. After that, when a piece of evidence e is observed, the posterior probability distribution of any variable of interest given e is obtained through inference on the Bayesian network, say $P(X_i|e)$. At last, users compare $P(X_i|e)$ and $P(X_i)$ in order to drawn a conclusion according to the difference between these two probability distributions.

We first propose two important theorems which lay the foundation of path propagation. Suppose there are two adjacent cliques C_i with its marginal $P(C_i)$ and C_j with its marginal $P(C_j)$. Their separator is S_{ij} ($S_{ij}=C_i\cap C_j$) with its marginal $P(S_{ij})$. When evidence e is observed such that $v(e)\subseteq C_i$, where $v(e)$ denotes all variables in e, then $P(C_i)$ will be updated to $P(C_i, e)$. Theorem 1 is proposed to pass e from C_i to C_j.

Theorem 1:

$$P(C_j,e)=P(C_j)\cdot P(e\,|\,S_{ij})=P(C_j)\cdot\frac{\sum_{C_i-v(e)-S_{ij}}P(C_i,e)}{P(S_{ij})}$$

Proof of Theorem 1:

First of all, according to $P(X,Y)=P(X)P(Y\mid X)=P(Y)P(X\mid Y)$ [5][7], the following equation is obtained:

$$P(C_j,e)=P(C_j)\cdot P(e\mid C_j). \tag{1}$$

Since $C_j=S_{ij}\cup(C_j-S_{ij})$, equation (1) can be rewritten as:

$$P(C_j,e)=P(C_j)\cdot P(e\mid C_j)=P(C_j)\cdot P(e\mid S_{ij},C_j-S_{ij}). \tag{2}$$

Due to the structure of the junction tree, since $v(e)\subseteq C_i$, e is conditional independent of (C_j-S_{ij}) given S_{ij} [9], that is, $P(e\mid S_{ij},C_j-S_{ij})=P(e\mid S_{ij})$. Thus, equation (2) can be converted to the following equation:

$$P(C_j,e)=P(C_j)\cdot P(e\mid C_j)=P(C_j)\cdot P(e\mid S_{ij},C_j-S_{ij})=P(C_j)\cdot P(e\mid S_{ij}). \tag{3}$$

Secondly, according to $P(X,Y)=P(X)P(Y\mid X)=P(Y)P(X\mid Y)$ [5][7], the following equation can be obtained:

$$P(e\mid S_{ij})=\frac{P(S_{ij},e)}{P(S_{ij})}. \tag{4}$$

Since $C_i=S_{ij}\cup(C_i-S_{ij})$, then

$$P(S_{ij})=\sum_{C_i-S_{ij}}P(C_i).$$

Similarly,

$$P(S_{ij},e)=\sum_{C_i-v(e)-S_{ij}}P(C_i,e). \tag{5}$$

Combine equation (4) with equation (5), then

$$P(e\mid S_{ij})=\frac{P(S_{ij},e)}{P(S_{ij})}=\frac{\sum_{C_i-v(e)-S_{ij}}P(C_i,e)}{P(S_{ij})}. \tag{6}$$

After combining equation (6) with equation (3), Theorem 1 is finally obtained:

$$P(C_j,e)=P(C_j)\cdot P(e\mid S_{ij})=P(C_j)\cdot\frac{\sum_{C_i-v(e)-S_{ij}}P(C_i,e)}{P(S_{ij})}.$$

Suppose there are two adjacent cliques C_i with its marginal $P(C_i,e_i)$ and C_j with its marginal $P(C_j,e_j)$. Assume e_i and e_j are two different pieces of compatible evidence associated to C_i and C_j. When new evidence e' is observed and $v(e')\subseteq C_i$,, $P(C_i,e_i)$ will be updated to $P(C_i,e_i,e')$. After that, C_i with its updated marginal $P(C_i,e_i,e')$ passes evidence e' to clique C_j.

Theorem 2:

The Clique C_i with its updated marginal $P(C_i,e_i,e')$ passing evidence e' to clique C_j correctly results in the updated marginal $P(C_j,e_j,e')$.

Proof of Theorem 2:

Actually, Theorem 2 can be similarly proved as Theorem 1. First of all, when e' is observed, the updated marginalized table $P(C_i,e_i)$ of C_i absorbs e' to the updated table $P(C_i,e_i,e')$. Secondly, e' can be passed from $P(C_i,e_i,e')$ to C_j associated with $P(C_j,e_j)$ to return $P(C_j,e_j,e')$ according to Theorem1. Therefore, e' can be successfully passed between two adjacent cliques associated with two different pieces of compatible evidence.

After introducing Theorem 1 and 2 of path propagation, we begin to discuss some important scenarios for queries in path propagation. In practice, queries have some query scenarios. In order to make them clear and concrete, take a query scenario in medical diagnosis as an example. If a doctor is diagnosing a patient with the help of a Bayesian network, the doctor will ask the patient to take a couple of clinical tests. Then the posterior probability distribution of a disease of interest given these clinical tests is obtained. However, the result shows that its posterior probability distribution does not vary too much to its prior probability distribution, which means that it is not justifiable for the doctor to conclude the diagnosis of this disease based on these clinical tests. Then the doctor switches to the posterior probability distribution of another disease given these clinical tests. This query scenario is formalized to scenario 1 of path propagation.

Scenario 1 (multiple variables): Given the evidence e, the posterior probability distributions for $\boldsymbol{X_1, X_2, \ldots, X_n}$ *given the fixed evidence e can be calculated,* *e.g.*, $P(X_1 \mid e), P(X_2 \mid e) \ldots, P(X_n \mid e)$.

Take another query scenario in medical diagnosis as an example. When a doctor diagnoses a patient with the help of a Bayesian network, the patient is required to take a couple of clinical tests. Then the posterior probability distribution of a disease of interest given these clinical tests is obtained. However, the result shows that its posterior probability distribution can not warrant the doctor to conclude the diagnosis of the disease based on these clinical tests until other clinical tests are taken to double confirm the conclusion. Then the doctor has to ask the patient to do some other clinical tests to return the posterior probability distribution of this disease of interest given all the clinical tests. This query scenario is formalized to scenario 2 of path propagation.

Scenario 2 (multiple evidences): Compute the posterior probability distributions for a fixed variable X of interest given these incremental pieces of evidence $e_1, e_2, \ldots, e_n$, *e.g.*, $P(X_1 \mid e_1), P(X_1 \mid e_1, e_2) \ldots, P(X_1 \mid e_1, \ldots, e_n)$, *where these pieces of evidence* $e_1, e_2, \ldots, e_n$ *should be compatible.*

However, in many real applications, there may contain more complex query scenarios. But they can be considered as the combination of scenario 1 and scenario 2. More details are introduced in the next section. In section 3.3, the prototype of path propagation based on Theorem 1 and 2 is proposed for computing queries.

3.3 THE PROTOTYPE OF PATH PROPAGATION

Consider a variable X and evidence e in a Bayesian network, where we assume that variables in e are all contained in a clique called C. According to the prototype of path propagation shown as follows, the posterior probability distribution of X given e, $P(X|e)$, can be obtained.

PROCEDURE Calculate(X, e)

Input: X and e such that variables in e are contained by a clique C.

```
Output: $P(X|e)$
{
1.  Identify a clique $C'$ such that $X \in C'$.
2.  Find out a path ($C_0 \to C_1 \to ... \to C_m$) in the junction tree such that $C_0 = C, C_m = C'$,
    and $C_{k-1} \to C_k$ represents an edge in the junction tree, where $k$=1,...,$m$. The symbol
    "$\to$" means the directionality of a path.
3.  For $k$=1 to $m$, clique $C_{k-1}$ passes $e$ to $C_k$ based on Theorem 1 and 2, which results in
    the updated marginal of each clique with $e$ in the path, e.g., $P(C_{k-1}, e)$.
4.  Calculate $P(X|e)$ from $P(C_m, e)$.
5.  Mark each clique $C_i$, $i$=0,...,$m$, with $e$, denoted as $C_i^e$.
6.  Return $P(X|e)$.
}
```

Take an example of how to calculate the posterior probability distribution of variable ***G*** given evidence ***A****=on* according to the prototype of path propagation in Figure 12. A clique containing evidence ***A*** is identified as the head node which is clique ***ABD*** and a clique containing variable ***G*** of interest as the tail node which is clique ***CEG***. Then a path between clique ***ABD*** and clique ***CEG*** is determined by the *Depth-first search (DFS) algorithm* [39]. Even though there are many existing algorithms for finding a path in an undirected tree, DFS searches deeper in the graph whenever possible. It starts at some node as the root in a tree and then explores as far as possible along each branch of the root before backtracking to the most recent node that it has not finished exploring [39]. Therefore, DFS is implemented in the thesis to identify a path in a junction tree. In this example, the path (clique ***ABD***→ clique ***ADE***→clique ***ACE***→clique ***CEG***) is identified. So evidence ***A****=on* can be passed along the path, starting at the head node and ending at the tail node. Arrows in Figure 12 illustrate the procedure of passing evidence ***A****=on*. After clique ***CEG*** receives the evidence, the posterior probability distribution of variable ***G*** given the evidence can be calculated from clique ***CEG*** immediately. Simultaneously, every clique in the path is marked with evidence ***A****=on* in order to demonstrate that marginal on each clique is *conjoint* with the evidence. Conjoint means the updated marginal of each clique with evidence A=*on* in the path is obtained after passing the evidence along the path.

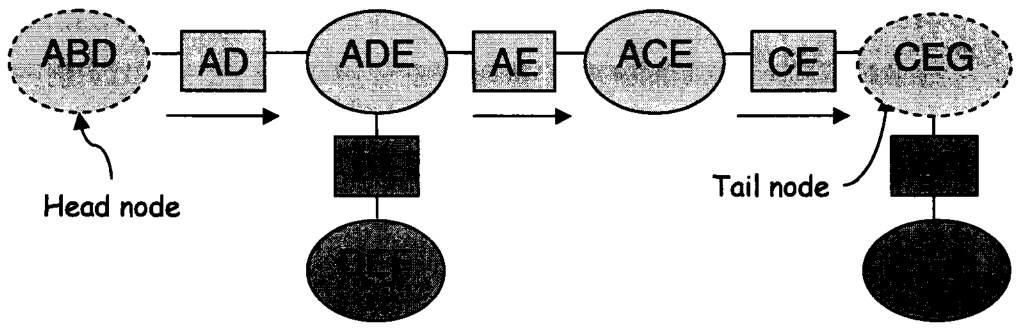


Figure 12: Applying the procedure of path propagation to return the posterior probability distribution of variable G given evidence A=*on*.

According to the prototype of path propagation, it is based on a query imposed by users and it only answers the query. It only carries out inference in the cliques included in a path. Other cliques which are not included in the path are not considered. Also, as a side effect of the prototype of path propagation, since the updated marginals of all cliques with the observed evidence in a path are calculated in step 3, if some variable contained in one of these cliques is interested, then the posterior probability distribution of this variable given the observed evidence is also readily obtained.

3.4 QUERY SCENARIOS OF PATH PROPAGATION

In this section, the analysis of scenario 1 and 2 is discussed at first and then how to solve more complex query scenarios is proposed.

3.4.1 Analysis of Scenario 1

In scenario 1, if a fixed evidence e is observed and a variable X_1 is interested, $P(X_1|e)$ can be successfully obtained through the procedure of path propagation, *Calculate*(X_1, e). After that, if another variable X_2 is interested given e, through *Calculate*(X_2, e), $P(X_2|e)$ can be also successfully obtained. Similarly, for more variables of interest (e.g., X_3, X_4, ..., X_n) given e, the posterior probability distributions of these variables given e can be returned by calling *Calculate*(X_i, e), where i=3 to n. There is an example that shows how the procedure of path propagation works for scenario 1.

Example 1:

Consider a simple Bayesian network. Suppose evidence ***D****=yes* is observed. How to return the posterior probability distribution of variable ***X*** of interest given evidence ***D****=yes* and later the posterior probability distribution of variable ***T*** of interest given evidence ***D****=yes*? The solution is to call *Calculate(****X****, ****D****=yes)* first and *Calculate(****T****, ****D****=yes)* thereinafter.

*Calculate(****X****, ****D****=yes):*

i. Two cliques are identified. One is clique 0 containing evidence ***D****=yes*. Another is clique 3 containing variable ***X***.
ii. In the junction tree, a path form clique 0 to clique 3 is identified by Depth-first Search algorithm. Thus, the path (clique 0→clique 4→clique 3) is determined.
iii. Evidence ***D****=yes* is passed along the path according to Theorem 1, starting from clique 0 and ending to clique 3.
iv. *P(****X****|****D****=yes)* is computed from clique 3.
v. Every clique in the path is marked with evidence ***D****=yes* in order to demonstrate that the updated marginal on the clique is conjoint with evidence ***D****=yes*.

Figure 13 illustrates the procedure of *Calculate(****X****, ****D****=yes)*. Arrows in Figure 13 indicate the path (clique 0→clique 4→clique 3) and evidence ***D****=yes* is marked on the cliques included in the path.

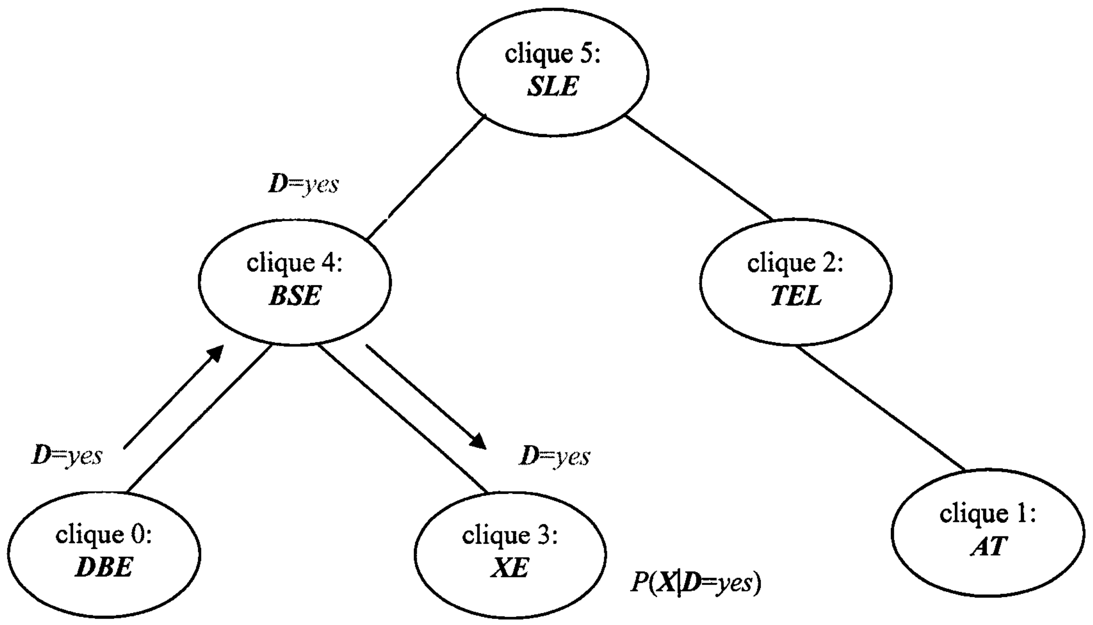


Figure 13: *Calculate(X, **D**=yes)* in scenario 1.

*Calculate(**T**, **D**=yes):*

i. We only identify clique 2 containing variable ***T*** of interest, since clique 0 containing evidence ***D**=yes* has already been identified during *Calculate(X, **D**=yes).*

ii. In the junction tree, the path form clique 0 to clique 2 is identified through Depth-first Search algorithm. Thus, the path (clique 0→clique 4→clique 5→clique 2) is determined.

iii. Evidence ***D**=yes* is passed along the path according to Theorem 1, starting from clique 0 and ending to clique 2.

iv. *P(**T**|**D**=yes)* is computed from clique 2.

v. Every clique in the path is marked with evidence ***D**=yes* in order to demonstrate that the updated marginal on the clique is conjoint with evidence ***D**=yes.*

However, from Figure 13, the updated marginals on clique 0 and clique 4 are already conjoint with evidence ***D**=yes* from *Calculate(X, **D**=yes).* Thus, evidence ***D**=yes* does not need to be passed again from clique 0 to clique 4 in *Calculate(**T**, **D**=yes).* In other words, evidence ***D**=yes* only needs to be passed from clique 4 to clique 2 through clique 5. In that case, the path (clique 0→clique 4→clique 5→clique 2) can be shortened to the path (clique 4→clique 5→clique 2). Figure 14 illustrates the procedure of *Calculate(**T**, **D**=yes)*. Arrows in Figure 14 indicate the path (clique 0→clique 4→clique 5→clique 2) and evidence ***D**=yes* is marked on the cliques which are conjoint with evidence ***D**=yes*.

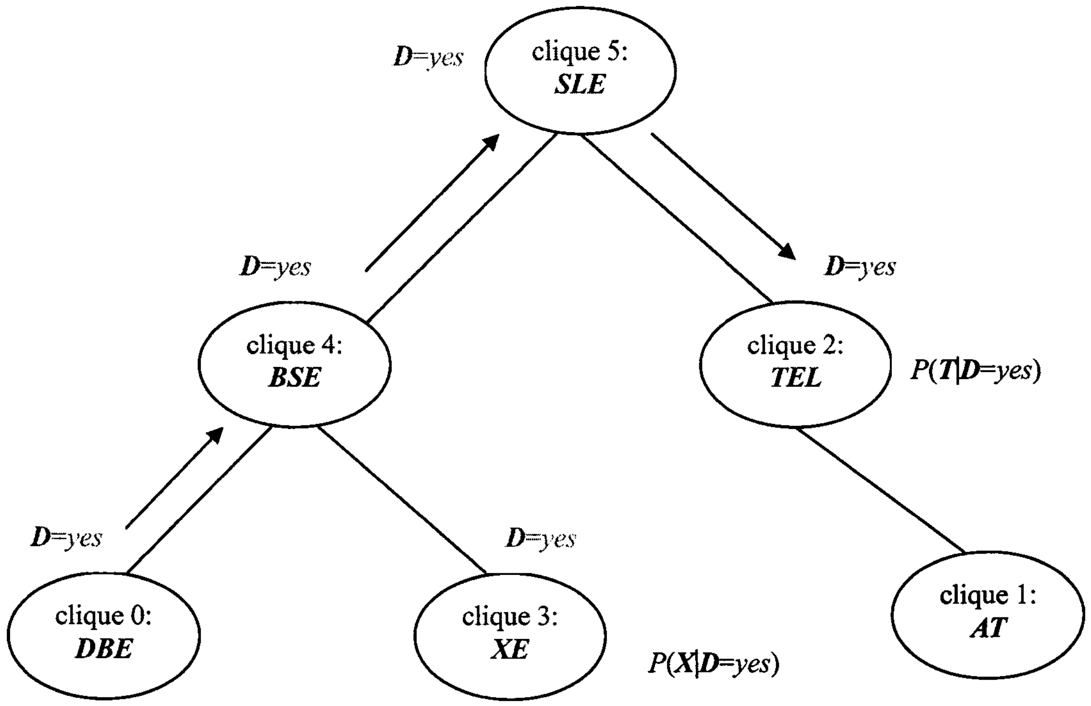


Figure 14: *Calculate(**T**, **D**=yes)* in scenario 1.

Regarding Example 1, a path between clique C_0 containing a variable $\boldsymbol{X}$ of interest and clique C_m where evidence e is originally residing can be possibly shortened. In other words, e may have been already updated by some cliques in the path during previous calculations. Thus, e only needs to be passed to the cliques that have not incorporated it. In step 2 of the procedure of path propagation, a path ($C_0 \rightarrow C_1 \rightarrow \ldots \rightarrow C_m$) is examined orderly from C_0 to C_m to find the last clique C_i ($0<i<m$) whose marginal has been updated with e. So the original path ($C_0 \rightarrow C_1 \rightarrow \ldots \rightarrow C_m$) can be shortened to path ($C_i \rightarrow C_{i+1} \rightarrow \ldots \rightarrow C_m$). In that case, more computations can be saved for calculating the posterior probability distribution of $\boldsymbol{X}$ given e. According to previous discussion, path propagation only involves a path in the junction tree to compute a query, so only a few cliques in the path participate in calculating the query.

3.4.2 Analysis of Scenario 2

In scenario 2, if evidence e_1 is observed and a fixed variable X is interested, $P(X| e_1)$ can be successfully obtained through the procedure of path propagation, *Calculate*(X, e_1') where $e_1'=e_1$. After that, if another compatible piece of evidence e_2 is observed based on the same variable of interest, through *Calculate*(X, e_2') where $e_2'=e_1+e_2$, $P(X|e_1, e_2)$ can be also successfully obtained. Similarly, for more compatible pieces of evidence observed (e.g., e_3, e_4, ..., e_n), the posterior probability distributions of X given them can be returned by calling *Calculate*(X, e_i'), where $e_i' = e_1+e_2+...+ e_i$ and i=3 to n. Solving scenario 2 is based on Theorem 1 & 2 and the prototype of path propagation. There is an example that shows how path propagation works for scenario 2.

Example 2:

Consider a simple Bayesian network. Suppose variable ***D*** is interested. How to return the posterior probability distribution of ***D*** given evidence ***X***=*yes*, *P*(***D***|***X***=*yes*)), and then the posterior probability distribution of ***D*** given evidence ***X***=*yes* and ***T***=*yes*, *P*(***D***|***X***=*yes*, ***T***=*no*))? The solution is to call *Calculate(**D**, **X**=yes)* first and *Calculate(**D**,(**X**=yes, **T**=no))* thereinafter.

*Calculate(**D**, **X**=yes):*

1. Two cliques are identified. One is clique 3 containing evidence ***X***=*yes*. Another is clique 0 containing variable ***D***.
2. In the junction tree, a path starting form clique 3 to clique 0 is identified by Depth-first Search Algorithm. Thus, the path (clique 3→clique 4→clique 0) is determined.
3. Evidence ***X***=*yes* is passed along the path according to Theorem 1, starting from clique 3 and ending to clique 0.
4. *P*(***D***|***X***=*yes*) is computed from clique 0.
5. Every clique in the path is marked with evidence ***X***=*yes* in order to demonstrate that the updated marginal on the clique is conjoint with evidence ***X***=*yes*.

Figure 15 illustrates the procedure of *Calculate(**D**, **X**=yes)*. Arrows in Figure 15 indicate the path (clique 3→clique 4→clique 0) and evidence ***X**=yes* is marked on the cliques included in the path.

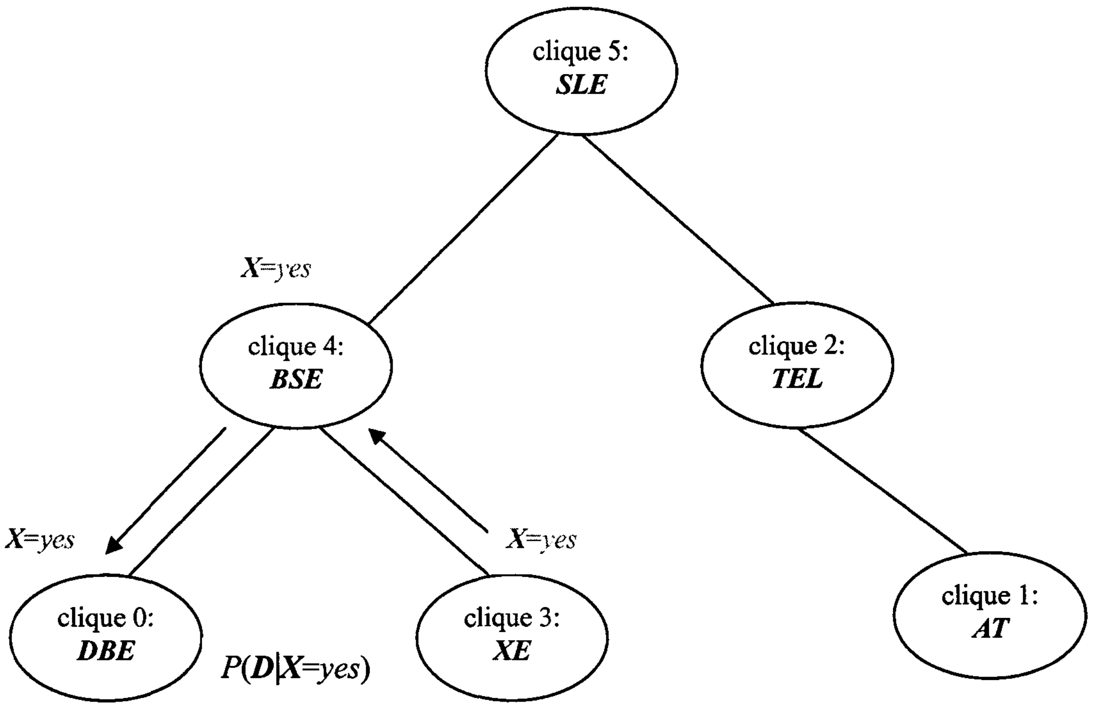


Figure 15: *Calculate(**D**, **X**=yes)* in scenario 2.

*Calculate(**D**, (**X**=yes, **T**=no)):*

i. We only identify clique 2 containing evidence ***T**=no*, since clique 0 including ***D*** of interest has already been identified during *Calculate(**D**, **X**=yes)*.

ii. In the junction tree, the path starting form clique 2 to clique 0 is identified through Depth-first Search Algorithm. Thus, the path (clique 2→clique 5→clique 4→clique 0) is determined.

iii. Evidence ***T**=no* is passed along the path according to Theorem 1 and 2, starting from clique 2 and ending to clique 0.

iv. *P(**D**|**X**=*yes, ***T***=no) is computed from clique 0.

v. Evidence ***T**=no* is marked to every clique in the path.

Arrows in Figure 16 indicate the path (clique 2→clique 5→clique 4→clique 0) and evidence *T=no* is marked on the cliques which are conjoint with evidence *X=yes*.

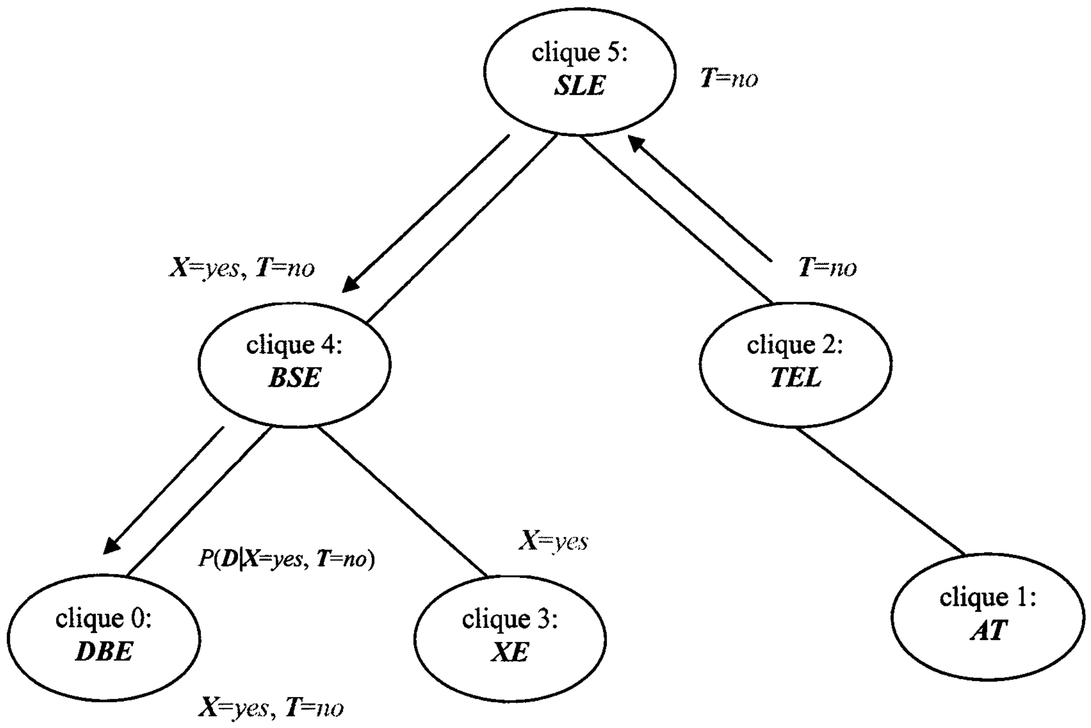


Figure 16: *Calculate(D, (X=yes, T=no))* in scenario 2.

Regarding Example 2, a clique containing the variable of interest is considered as the root clique. Whenever a piece of new evidence is observed, we pass it from a clique containing the new evidence to the root clique through a path identified in the junction tree. And mark the cliques with the new evidence in the path so that the updated marginal of the root clique is always conjoint with all pieces of new evidence which are observed so far. According to previous discussion, path propagation only involves a path in the junction tree to compute the posterior probability distribution of a fixed variable of interest given one piece of evidence. Whenever a piece of new evidence is observed, a new path will be identified by path propagation and inference will be carried out in the new path. However, for global propagation, a full scale global propagation is carried out on the whole junction tree for the posterior probability distribution of a fixed variable of interest given one piece of evidence. Whenever a piece of new evidence is observed, another full scale global propagation is carried out on the whole junction tree.

3.4.3 Analysis of Complex Query Scenarios

Actually, in many real applications, there may exhibit more complex query scenarios. However, they can be considered as the combination of scenario 1 and scenario 2. Take a query scenario in medical diagnosis as an example. If a doctor is diagnosing a patient with the help of a Bayesian network, the doctor will ask the patient to take a couple of clinical tests e_1. Then the posterior probability distribution of a disease X of interest given these clinical tests is obtained, $P(X| e_1)$. However, the result shows that $P(X| e_1)$ does not vary too much to its prior probability distribution $P(X)$, which means that it is not justifiable for the doctor to conclude the diagnosis of this disease based on these clinical tests. Then the doctor switches to the posterior probability distribution of another disease Y of interest given e_1. Unfortunately, the result shows that $P(Y| e_1)$ can not warrant the doctor to conclude the diagnosis of Y given e_1 until other clinical tests are taken to double confirm the conclusion. Then the doctor has to ask the patient to do some other clinical tests e_2 to see the posterior probability distribution of Y given e_1 and e_2, $P(Y| e_1,e_2)$. This query scenario can not be formalized to scenario 1 or scenario 2 but it is actually the combination of scenario 1 and 2. During the procedure of calculating $P(X| e_1)$ and $P(Y| e_1)$, it can be considered as scenario 1. Then the remaining procedure of calculating $P(Y| e_1,e_2)$ from $P(Y| e_1)$ can be regarded as scenario 2. Hence, for complex query scenarios, the posterior probability distributions can be easily obtained according to scenario 1 and 2, that is, they can be actually regarded as interweaving cases of scenario 1 and 2. Therefore, scenario 1 and 2 are two basic query scenarios for path propagation. Hence, if the computational efficiency of path propagation for scenario 1 and 2 is better than that of global propagation, then it will be also better than that of global propagation in more complex query scenarios. So in the next chapter, experiments of path propagation and global propagation are carried out for scenario 1 and 2.

CHAPTER IV

IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this chapter, experiments are carried out for both path propagation and global propagation. First of all, the basic implementation information is provided, such as the implementation environment, and the implementation procedure for path propagation and global propagation. Secondly, the experimental results for these two propagation methods are presented. Finally, some conclusions are drawn based on these experimental results.

4.1 THE INFORMATION OF IMPLEMENTATION

In the thesis, we choose the C programming language under Unix to implement path propagation and global propagation. The main reason is that the Hugin Tool is developed by the C programming language under Unix [47]. There are more reasons listed as follows:

i. The C programming language is a relatively minimalistic, general-purpose, imperative, procedural programming language under Unix. It could be compiled in a straightforward manner using a relatively simple compiler, provide low-level access to memory, generate only a few machine language instructions for each of its core language elements, and not require extensive run-time support [40].

ii. As far as the C programming language is concerned, sine it was originally developed under Unix, it remains very popular in Unix world.

In order to compare the computational efficiency of path propagation and global propagation, they should be implemented based on the same comparison condition. First of all, the data structure used in path propagation is identical as the data structure used in global propagation. Secondly, methods of computations (e.g., multiplication of two probabilities or potentials, marginalization of a probability or potential into the marginalized table, normalization of two probabilities or potentials, division of two probabilities or potentials, absorption of pieces of evidence into a probability or potential), are all exactly the same. According to Chapter 3, the only difference between path propagation and global propagation is that path propagation carries out inference only in certain paths in a junction tree. The following steps briefly introduce the implementation procedure.

Step 1: Convert a Bayesian network into the data structure of a DAG

Since a Bayesian network contains two components, a DAG and the CPTs, we use a *.net file* [47] to completely store these two components of a Bayesian network. As a representation of Bayesian networks, a .net file contains the whole information of every variable (e.g., its label, its position, and its states) in a Bayesian network and its CPT. In this thesis, a .net file is parsed into the data structure of a DAG. In this data structure, every variable with its associated CPT is stored. In this step, the implementation methods for path propagation and global propagation are identical.

Step 2: Triangulate a Bayesian network into a junction tree

The data structure of a DAG is converted into the data structure of a junction tree. The way for triangulation into a junction tree introduced in Section 2.3.1 is implemented for both path propagation and global propagation. Hence, in this step, the implementation methods for path propagation and global propagation are exactly the same.

Step 3: Initialize a potential for each clique in the junction tree

After triangulation, the CPTs are assigned into corresponding cliques by following the standard criteria mentioned in Section 2.3.1. Then for each clique in the junction tree, the CPTs assigned to it are multiplied to be a potential for this clique. The implementation methods for path propagation and global propagation are identical in this step.

Step 4: Inference

In this step, the implementation methods for path propagation and global propagation are different. According to Chapter 3, path propagation has two basic query scenarios which are scenario 1 and 2. Other complex query scenarios can be solved by the combination of scenario 1 and 2. Hence, if the computational efficiency of path propagation for scenario 1 and 2 is better than that of global propagation, then it will be also better than that of global propagation in more complex query scenarios. In that case, scenario 1 and 2 are implemented by these two propagation methods in the thesis.

Scenario 1 (multiple variables):

For global propagation, a full scale global propagation with evidence observed on the whole junction tree is applied according to Section 2.3.4. Then all cliques and separators in the junction tree are marginalized. In order to return the posterior probability distributions of multiple variables given the fixed evidence, some separators or cliques containing the variables of interest are identified and normalization is applied on them. For path propagation, a path between a clique containing the variable of interest and another clique containing the evidence is identified. Path propagation is applied on this path. Regarding Section 3.3.1, different paths on the junction tree are identified to return the posterior probability distributions of multiple variables given the fixed evidence.

Scenario 2 (multiple evidences):

For global propagation, according to Section 2.3.4, when a piece of evidence is observed, a full scale global propagation on the whole junction tree is applied. If another new piece of evidence is observed, another full scale global propagation on the whole junction tree is applied. However, for path propagation, regarding Section 3.3.2, different paths on the whole junction tree are identified to obtain the posterior probability distributions of the fixed variable of interest given multiple evidences.

In summary, Table 1 illustrates the implementation procedure for path propagation and global propagation. In Table 1, SAME represents identical methods while DIFFERENT represents different methods used in the implementation of these two propagation methods.

Implementation Procedure		Path propagation	Global propagation
Step 1: Convert a Bayesian network into the data structure of a DAG		SAME	SAME
Step 2: Triangulate a Bayesian network into a junction tree		SAME	SAME
Step 3: Initialize a potential for each clique in the junction tree		SAME	SAME
Step 4: Inference	Scenario 1	DIFFERENT	DIFFERENT
	Scenario 2	DIFFERENT	DIFFERENT

Table 1: The implementation procedure for path propagation and global propagation.

4.2 EXPERIMENTAL RESULTS

After a brief introduction of the implementation procedure, experiments are carried out to compare the computational efficiency between path propagation and global propagation. In the thesis, several Bayesian networks are randomly chosen for the comparison between path propagation and global propagation. All these Bayesian networks can be downloaded from http://oldwww.cs.aau.dk/research/MI/Misc/networks.html, http://www.cs.huji.ac.il/labs/compbio/Repository/networks.html, and http://www.hugin.com/Products_Services/Products/Demo/. The experimental results of five Bayesian networks are described in this section. The sizes and nodes of these five Bayesian networks are gradually increasing. The first Bayesian network is called *Asia.net* with 8 nodes. After converting it to the junction tree, the total number of cliques for *Asia.net* is 6. The second Bayesian network is called *Car_ts.net* with 12 nodes. The junction tree for *Car_ts.net* includes 6 cliques. The third Bayesian network is called *4sp.net* with 58 nodes. 40 cliques are obtained from the junction tree for *4sp.net*. The fourth Bayesian network is called *Pigs.net* with 441 nodes and the number of total cliques in its resulted junction tree is 368. The fifth Bayesian network is called *Munin2.net* with 1003 nodes. The junction tree converted from *Munin2.net* includes 866 cliques.

From the traditional perspective, in order to compare computational efficiencies between two algorithms, the worst case complexities of these two methods have to be compared. However, there is no difference between path propagation and global propagation since their worst case complexities are both NP-hard as we mentioned before. In 1.[1], the number of binary arithmetic operations (denoted as # binary arithmetic operations) is used to measure the computational efficiency. It contains the number of additions, multiplications, and divisions. In the thesis, we also use # binary arithmetic operations to measure the computational efficiency. Since path propagation and global propagation are both based on the same comparison condition introduced in Section 4.1, if one propagation method causes more # binary arithmetic operations than the other one, then it is less computationally efficient than the other one. In the thesis, # binary arithmetic operations are only recorded during the different step which is step 4. The experimental results for scenario 1 and 2 are shown in the next two sections. For each of the five Bayesian networks, some query experiments of scenario 1 (or 2) are designed. A query experiment of scenario 1 (or 2) means a sequence of queries which exactly satisfies the definition of scenario 1 (or 2) in the thesis. These query experiments are independent of each other. All variables of interest and pieces of evidence are generated randomly in these query experiments.

4.2.1 Experimental Results for Scenario 1

The experimental results of the five Bayesian networks for scenario 1 are shown as follows. Tables shown in this section record the number of additions, multiplications, divisions, which are caused by path propagation and global propagation in each query experiment. Figures shown in this section demonstrate the comparison of the total number of calculations needed by these two propagation methods in each query experiment. X-axis in each figure represents the number of variables of interest in each query experiment while the numbers in Y-axis represent the total number of calculations caused by both two propagation methods.

Asia.net:

Three different query experiments are designed. The first query experiment only queries one variable of interest given *e*. The second query experiment incrementally queries three different variables of interest given *e*. The third query experiment incrementally queries five different variables of interest given *e*. Table 2 and Figure 17 illustrate the comparison of these three query experiments.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	22	21	21	64
	PATH PROPAGATION	16	15	15	46
Query experiment 2	GLOBAL PROPAGATION	26	21	23	70
	PATH PROPAGATION	20	15	17	52
Query experiment 3	GLOBAL PROPAGATION	30	21	25	76
	PATH PROPAGATION	25	16	20	61

Table 2: The comparison of # binary arithmetic operations of *Asia.net* for scenario 1.

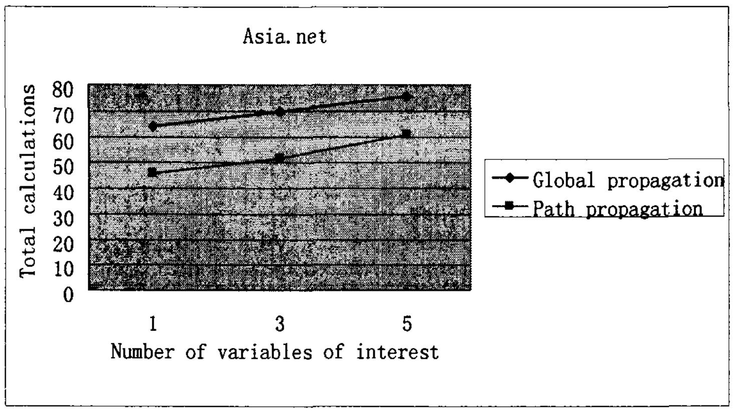


Figure 17: The comparison of # total calculations of *Asia.net* for scenario 1.

Car_ts.net:

Four different query experiments are designed. The first query experiment only queries one variable of interest given *e*. The second query experiment incrementally queries three different variables of interest given *e*. The third query experiment incrementally queries five different variables of interest given *e*. The forth query experiment incrementally queries seven different variables of interest given *e*. Table 3 and Figure 18 illustrate the comparison of these four query experiments.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	22	21	21	64
	PATH PROPAGATION	15	14	14	43
Query experiment 2	GLOBAL PROPAGATION	25	21	22	68
	PATH PROPAGATION	19	14	16	49
Query experiment 3	GLOBAL PROPAGATION	30	21	25	76
	PATH PROPAGATION	23	14	18	55
Query experiment 4	GLOBAL PROPAGATION	34	21	27	82
	PATH PROPAGATION	28	15	21	64

Table 3: The comparison of # binary arithmetic operations of *Car_ts.net* for scenario 1.

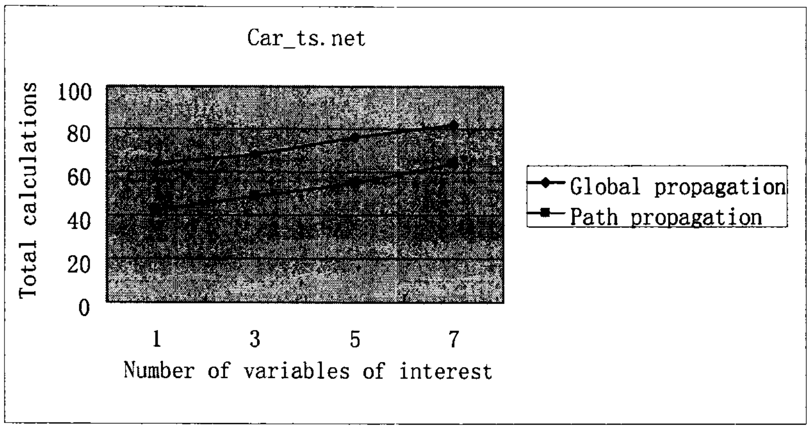


Figure 18: The comparison of # total calculations of *Car_ts.net* for scenario 1.

4sp.net:

Four different query experiments are designed. The first query experiment only queries one variable of interest given *e*. The second query experiment incrementally queries four different variables of interest given *e*. The third query experiment incrementally queries seven different variables of interest given *e*. The forth query experiment incrementally queries ten different variables of interest given *e*. Table 4 and Figure 19 illustrate the comparison of these four query experiments.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	162	164	161	487
	PATH PROPAGATION	87	86	86	259
Query experiment 2	GLOBAL PROPAGATION	168	164	164	496
	PATH PROPAGATION	104	97	100	301
Query experiment 3	GLOBAL PROPAGATION	174	164	167	505
	PATH PROPAGATION	117	104	110	331
Query experiment 4	GLOBAL PROPAGATION	180	164	170	514
	PATH PROPAGATION	129	110	119	358

Table 4: The comparison of # binary arithmetic operations of *4sp.net* for scenario 1.

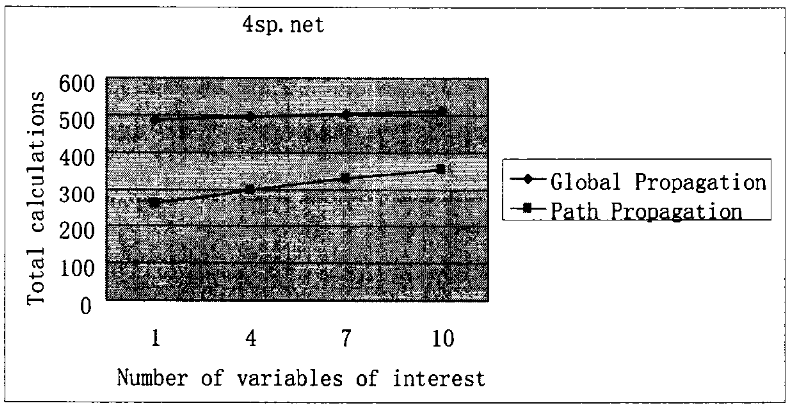


Figure 19: The comparison of # total calculations of *4sp.net* for scenario 1.

Pigs.net:

Five different query experiments are designed. The first query experiment only queries one variable of interest given *e*. The second query experiment incrementally queries two different variables of interest given *e*. The third query experiment incrementally queries three different variables of interest given *e*. The forth query experiment incrementally queries four different variables of interest given *e*. The fifth query experiment incrementally queries five different variables of interest given *e*. The experimental results are illustrated in Table 5 and Figure 20.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	1470	1469	1469	4408
	PATH PROPAGATION	754	753	753	2260
Query experiment 2	GLOBAL PROPAGATION	1472	1469	1470	4411
	PATH PROPAGATION	761	758	759	2278
Query experiment 3	GLOBAL PROPAGATION	1474	1469	1471	4414
	PATH PROPAGATION	763	758	760	2281
Query experiment 4	GLOBAL PROPAGATION	1476	1469	1472	4417
	PATH PROPAGATION	773	766	769	2308
Query experiment 5	GLOBAL PROPAGATION	1478	1469	1473	4420
	PATH PROPAGATION	781	772	776	2329

Table 5: The comparison of # binary arithmetic operations of *Pigs.net* for scenario 1.

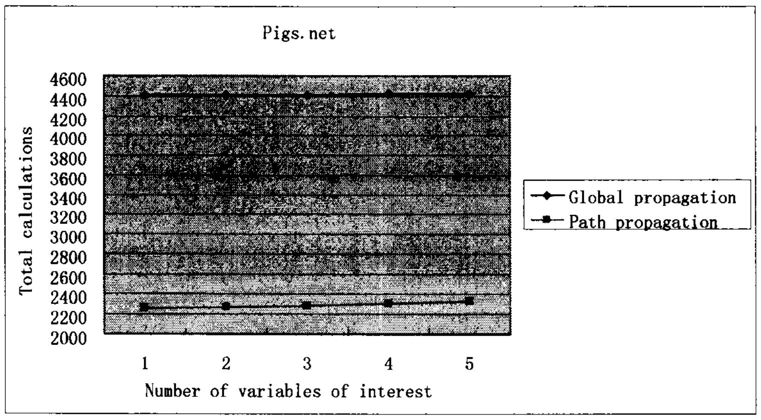


Figure 20: The comparison of # total calculations of *Pigs.net* for scenario 1.

Munin2.net:

Three different query experiments are designed. The first query experiment only queries one variable of interest given *e*. The second query experiment incrementally queries two different variables of interest given *e*. The third query experiment incrementally queries three different variables of interest given *e*. The experimental results are illustrated in Table 6 and Figure 21.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	3462	3463	3461	10386
	PATH PROPAGATION	1744	1743	1743	5230
Query experiment 2	GLOBAL PROPAGATION	3464	3463	3462	10389
	PATH PROPAGATION	1753	1750	1751	5254
Query experiment 3	GLOBAL PROPAGATION	3466	3463	3463	10392
	PATH PROPAGATION	1767	1762	1764	5293

Table 6: The comparison of # binary arithmetic operations of *Munin2.net* for scenario 1.

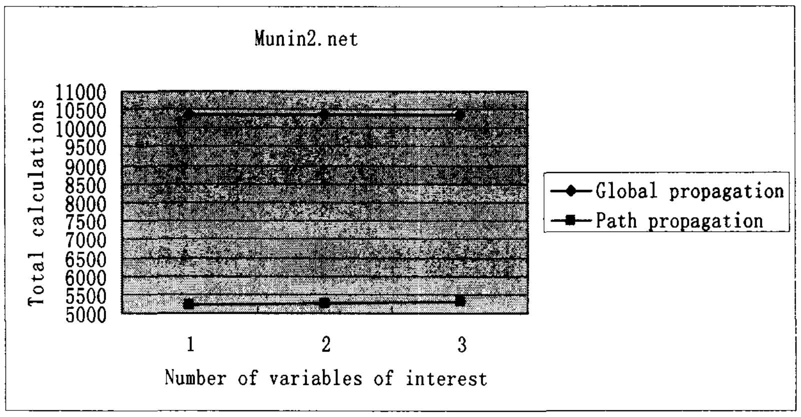


Figure 21: The comparison of # total calculations of *Munin2.net* for scenario 1.

4.2.3 Experimental Results for Scenario 2

The experimental results of the five Bayesian networks for scenario 2 are shown as follows. Tables shown in this section record the number of additions, multiplications, divisions, which are caused by path propagation and global propagation in each query experiment. Figures shown in this section demonstrate the comparison of the total number of calculations needed by these two propagation methods in each query experiment. X-axis in each figure represents the number of variables in all pieces of evidence in each query experiment while the numbers in Y-axis represent the total number of calculations caused by both two propagation methods.

Asia.net:

Four different query experiments are designed. The first query experiment only queries the fixed variable of interest given one piece of evidence containing one variable. The second query experiment queries the fixed variable of interest given two incremental pieces of evidence and each of these two pieces of evidence contains one variable. The third query experiment queries the fixed variable of interest given three incremental pieces of evidence and each of these three pieces of evidence contains one variable. The fourth query experiment queries the fixed variable of interest also given three incremental pieces of evidence. But the first two pieces of evidence both contain two variables and the last piece of evidence contains one variable. Table 7 and Figure 22 illustrate the comparison of these four query experiments.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	22	22	21	65
	PATH PROPAGATION	16	15	15	46
Query experiment 2	GLOBAL PROPAGATION	34	39	32	105
	PATH PROPAGATION	20	18	18	56
Query experiment 3	GLOBAL PROPAGATION	46	58	43	147
	PATH PROPAGATION	25	22	22	69
Query experiment 4	GLOBAL PROPAGATION	46	49	43	138
	PATH PROPAGATION	22	21	19	62

Table 7: The comparison of # binary arithmetic operations of *Asia.net* for scenario 2.

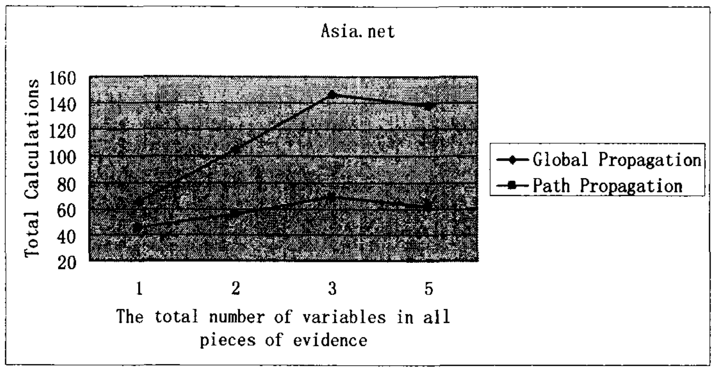


Figure 22: The comparison of # total calculations of *Asia.net* for scenario 2.

Car_ts.net:

Four different query experiments are designed. The first query experiment only queries the fixed variable of interest given one piece of evidence containing one variable. The second query experiment queries the fixed variable of interest given two incremental pieces of evidence and each of these two pieces of evidence contains one variable. The third query experiment queries the fixed variable of interest given three incremental pieces of evidence. The first piece of evidence contains one variable and the last two pieces of evidence contain two variables. The fourth query experiment queries the fixed variable of interest given four incremental pieces of evidence. The first piece of evidence contains one variable and the last three pieces of evidence all contain two variables. Table 8 and Figure 23 illustrate the comparison of these four query experiments.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	22	22	21	65
	PATH PROPAGATION	13	12	12	37
Query experiment 2	GLOBAL PROPAGATION	34	34	32	100
	PATH PROPAGATION	15	13	13	41
Query experiment 3	GLOBAL PROPAGATION	46	48	43	137
	PATH PROPAGATION	18	17	15	50
Query experiment 4	GLOBAL PROPAGATION	58	62	54	174
	PATH PROPAGATION	23	22	19	64

Table 8: The comparison of # binary arithmetic operations of *Car_ts.net* for scenario 2.

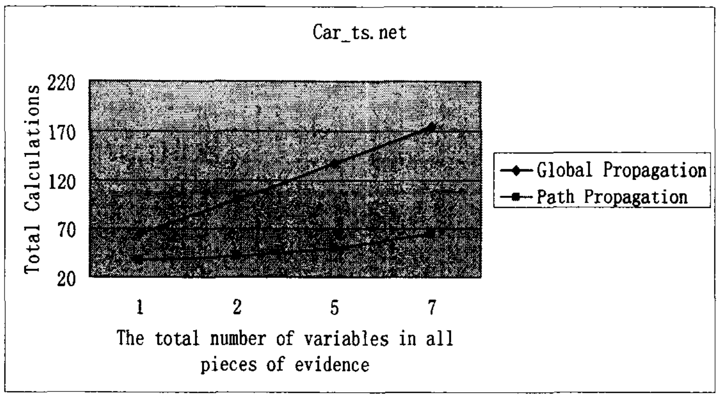


Figure 23: The comparison of # total calculations of *Car_ts.net* for scenario 2.

4sp.net:

Five different query experiments are designed. The first query experiment only queries the fixed variable of interest given one piece of evidence containing one variable. The second query experiment queries the fixed variable of interest given two incremental pieces of evidence and each of these two pieces of evidence contains one variable. The third query experiment queries the fixed variable of interest given three incremental pieces of evidence and each of these three pieces of evidence contains two variables. The forth query experiment queries the fixed variable of interest given four incremental pieces of evidence and each of these four pieces of evidence contains two variables. The fifth query experiment queries the fixed variable of interest given five incremental pieces of evidence and each of these five pieces of evidence contains two variables. Table 9 and Figure 24 illustrate the comparison of these five query experiments.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	162	164	161	487
	PATH PROPAGATION	87	86	86	259
Query experiment 2	GLOBAL PROPAGATION	244	247	242	733
	PATH PROPAGATION	95	93	93	281
Query experiment 3	GLOBAL PROPAGATION	326	331	323	980
	PATH PROPAGATION	109	109	106	324
Query experiment 4	GLOBAL PROPAGATION	408	413	404	1255
	PATH PROPAGATION	119	119	115	353
Query experiment 5	GLOBAL PROPAGATION	490	498	485	1473
	PATH PROPAGATION	121	. 121	116	358

Table 9: The comparison of # binary arithmetic operations of *4sp.net* for scenario 2.

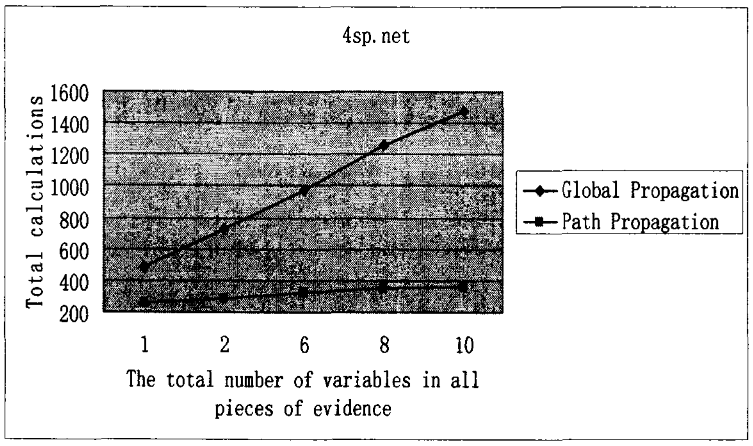


Figure 24: The comparison of # total calculations of *4sp.net* for scenario 2.

Pigs.net:

Three different query experiments are designed. The first query experiment only queries the fixed variable of interest given one piece of evidence containing one variable. The second query experiment queries the fixed variable of interest given two incremental pieces of evidence. The first piece of evidence contains one variable and the second piece of evidence contains two variables. The third query experiment queries the fixed variable of interest given three incremental pieces of evidence. The first piece of evidence contains one variable and the second piece of evidence contains two variables. The last piece of evidence contains three variables. The experimental results are illustrated in Table 10 and Figure 25.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	1470	1469	1469	4408
	PATH PROPAGATION	754	753	753	2260
Query experiment 2	GLOBAL PROPAGATION	2206	2230	2204	6640
	PATH PROPAGATION	772	771	770	2313
Query experiment 3	GLOBAL PROPAGATION	2942	2981	2939	8862
	PATH PROPAGATION	787	787	784	2358

Table 10: The comparison of # binary arithmetic operations of *Pigs.net* for scenario 2

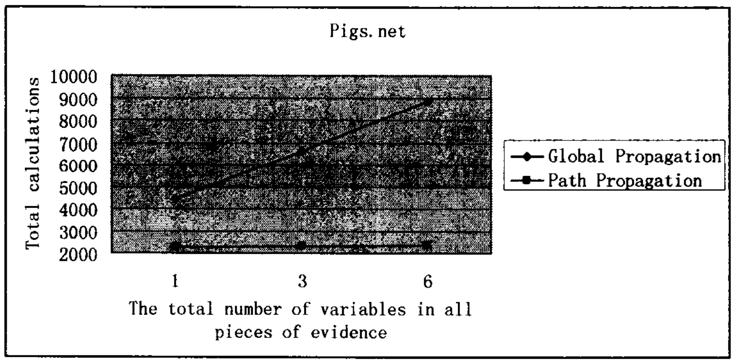


Figure 25: The comparison of # total calculations of *Pigs.net* for scenario 2.

Munin2.net:

Three different query experiments are designed. The first query experiment only queries the fixed variable of interest given one piece of evidence containing one variable. The second query experiment queries the fixed variable of interest given two incremental pieces of evidence. The first piece of evidence contains one variable and the second piece of evidence contains two variables. The third query experiment queries the fixed variable of interest given three incremental pieces of evidence. The first piece of evidence contains one variable and the second piece of evidence contains two variables. The last piece of evidence contains three variables. The experimental results are illustrated in Table 11 and Figure 26.

		# Additions	# Multiplications	# Divisions	# total calculations
Query experiment 1	GLOBAL PROPAGATION	3462	3463	3461	10386
	PATH PROPAGATION	1744	1743	1743	5230
Query experiment 2	GLOBAL PROPAGATION	5194	5226	5192	15612
	PATH PROPAGATION	1767	1766	1765	5298
Query experiment 3	GLOBAL PROPAGATION	6926	6989	6923	20838
	PATH PROPAGATION	1788	1788	1785	5361

Table 11: The comparison of # binary arithmetic operations of *Munin2.net* for scenario 2.

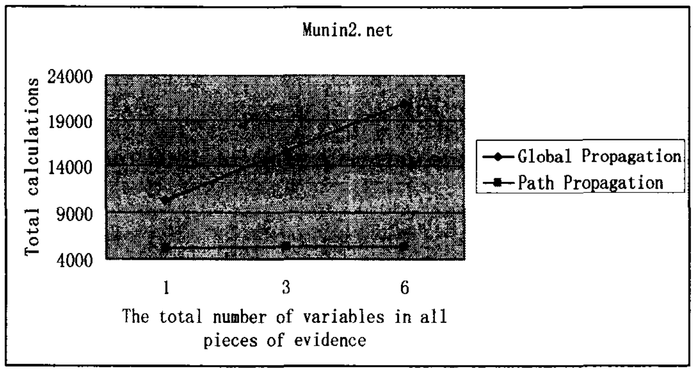


Figure 26: The comparison of # total calculations of *Munin2.net* for scenario 2.

4.3 EXPERIMENTAL EVALUATIONS

From the tables and figures presented in Section 4.2, several conclusions are drawn as follows.

Evaluations of the experimental results for scenario 1:

According to the figures and tables shown in Section 4.2.2, the number of calculations for path propagation is smaller than that of global propagation, while the former increases faster than the latter. The reason is that after a full scale global propagation on a junction tree with evidence observed once, the posterior probability distribution of any variable of interest given the fixed evidence can be obtained for global propagation. However, for path propagation, multiple paths are identified for the posterior probability distributions of multiple variables given the fixed evidence, which means more calculations are required by these paths for path propagation than global propagation. When users interest in more variables, the number of calculations for path propagation may become bigger than that of global propagation. In that case, the number of calculations for global propagation is only marginally bigger than that of global propagation. Hence, the curves of these two propagation methods in the figures may have an intersection. However, the x-axis value at the intersection representing the number of variables of interest will be very large due to the increasing sizes and nodes of Bayesian networks from the figures in Section 4.2.2. Moreover, since users often interest in the posterior probability distributions of a few variables in practice as we have mentioned before, the intersection does not concern to us. As the results shows, when Bayesian networks become large and complex, path propagation is more computationally efficient than global propagation for a few queries.

Furthermore, the objective for global propagation is that the posterior probability distribution of any variable of interest given the evidence in Bayesian networks can be obtained after a full scale global propagation. However, since users often interest a few variables in real applications, global propagation certainly wastes lots of computational resources especially in large and complex Bayesian networks. For path propagation, it is based on the query imposed by users and it answers the query only, so it takes less computational resources than global propagation.

Evaluations of the experimental results for scenario 2:

According to the tables and figures in Section 4.2.3, the number of calculations for path propagation is much smaller than that of global propagation. As the figures show, for global propagation, the number of calculations increases to approximately two times the number of previous calculations whenever new evidence is observed. Because when a piece of evidence is observed, a full scale global propagation will be performed on a junction tree; when another new piece of evidence is observed, another full scale global propagation will be performed on the whole junction tree; so on and so forth. However, for path propagation, the number of calculations increases much slowly whenever new evidence is observed. Because when a piece of evidence is observed, calculations will be incurred by a path in the junction tree; when another new piece of evidence is observed, calculations will only be incurred by another new path for the new evidence; so on and so forth. Hence, path propagation is more computationally efficient than global propagation.

Furthermore, for global propagation, after a full scale global propagation, the posterior probability distribution of any variable of interest given the evidence in Bayesian networks can be obtained. However, path propagation is based on a query imposed by users and it answers the query only, so it takes less computational resources than global propagation.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

The recent trend is to extend Bayesian networks into large and complex Bayesian networks [25]. Since the effectiveness and efficiency of global propagation are strongly related to the size of a Bayesian network in reality, when it become large and complex, global propagation is no longer suitable for inference. According to a failed experiment with the Hugin tool introduced in Section 3.1, so global propagation needs to be revised and improved. Path propagation is proposed for inference in large and complex Bayesian networks. It carries out inference only in certain paths in a junction tree. Also, it is based on a query imposed by users and answers the query only. It takes less time and computational resources to answer queries than global propagation especially in large and complex Bayesian networks.

Path propagation can be also improved in the following ways for future work.

i. In this thesis, all variables contained in any piece of evidence are assumed to be included in one clique. However, if there is a piece of evidence that all variables contained in the evidence are not included in any clique in a junction tree, then it will be decomposed into several smaller pieces of the evidence. All variables included in each of these smaller pieces of the evidence should be contained in a clique. For example, consider a piece of evidence e that includes two variables X and Y, where X=1 and Y=0. If X and Y are not included in any clique in the junction tree, then e has to be decomposed into two smaller pieces of evidence e' and e'', where e' denotes $X=1$ and e'' denotes $Y=0$. Thus, $v(e')$ and $v(e'')$ can be contained in a clique because e' and e'' have become singleton sets after the decomposition.

ii. According to the assumption of path propagation, a full scale global propagation should be performed once on the whole junction tree with no evidence observed to make every clique and separator marginalized. Since global propagation may fail to operate in large and complex Bayesian networks as the failed experiment shown in Section 3.1. A recent developed method implements global propagation as a database application. Since this method is combined with relational database, so it is determined only by the capacity of the database management system (DBMS) [11] not the capacity of internal memory [41][42][43]. Thus, this method can be applied to path propagation to solve the problem caused by the capacity of internal memory.

iii. Recently, an algorithm called lazy propagation is proposed. Lazy propagation, based on lazy evaluation, is quite useful in message passing. The time and space costs for performing lazy propagation on the Hugin architecture are smaller than that of global propagation [13]. Path propagation can be combined with lazy propagation. According to the assumption of path propagation, lazy propagation can be performed instead of global propagation on the whole junction tree once to get each clique and separator marginalized. Lazy propagation does not calculate the final marginalized potential for each clique and separator but just keeps the list of these potentials for them. It maintains a multiplicative decomposition of clique and separator potentials and postpones combination of potentials [13]. When a piece of evidence is observed and users interest in a variable, path propagation can be applied on the junction tree to identify a path and pass the evidence along the path to return the posterior probability distribution of the variable of interest given the evidence.

BIBLIOGRAPHY

[1] Vasilica Lepar and Prakash P. Shenoy, A Comparison of Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer Architectures for Computing Marginals of Probability Distributions, In G. Cooper and S. Moral, editors, Proc. of the Conf. on Uncertainty in AI, pages 328-337, Morgan Kaufmann, 1998.

[2] Bangso, O., and Wuillemin, P., Top-down construction and repetitive structures representation in Bayesian networks, In James N. Etheredge, B. Z. M., ed., Proceedings of the Thirteenth Internatinal Florida Artificial Intelligence Research Society Conference, 282-286, AAA Press, 2000.

[3] A Brief Introduction to Graphical Models and Bayesian Networks, http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html, 2001.

[4] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, An Introduction to Variational Methods for Graphical Models, Learning and Inference in Graphical Models, 37 (2): 75, 1998.

[5] Huang, C., and Darwiche, A., Inference in Belief networks: A procedure guide. International Journal of Approximate Reasoning 15(3):225-263, 1996.

[6] Technical Report: Basics of Bayesian Networks, http://www.agena.co.uk, 2002.

[7] Technical Report: Basics of Bayesian Probability, http://www.agena.co.uk, 2002.

[8] E. Charniak, Bayesian Networks without Tears, AI Magazine 12(4), 1991.

[9] Yang Xiang, Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach, Cambridge University Press, 2004.

[10] S. Lauritzen and D. Spiegelhalter, Local Computation with Probabilistic on Graphical Structures and Their Application to Expert Systems, Journal of the Royal Statistical Society, 50:157-244, 1988.

[11] S. K. M. Wong, D. Wu, and C.J. Butz, Probabilistic Reasoning in Bayesian Networks: a Relational Database Approach, Sixteenth Canadian Conference on Artificial Intelligence (AI), Pages: 583-590, 2003.

[12] F. Gagliardi Cozman, Generalizing Variable Elimination in Bayesian Networks, Technical Report of the Escola Politecnica, University of Sao Paulo, 2000.

[13] Anders L. Madsen, Finn V. Jensen, Lazy Propagation: A Junction Tree Inference Algorithm Based on Lazy Evaluation, Artificial Intelligence 113(1999) 203-245, 1999.

[14] S.L. Lauritzen and D.J. Spiegelhalter, Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems, J. Royal Statistical Society B, 50:157-224, 1988.

[15] R.A. Howard and J.E. Matheson, Influence Diagrams, In R.A. Howard and J.E. Matheson, editors, Readings on the Principles and Applications of Decision Analysis, volume II, pages 721-762, Strategic Decisions Group, Meno Park, CA., 1981.

[16] S. Olmsted, On Representing and Solving Decision Problems, PhD thesis, Department of Engineering-Economic Systems, Stanford University, 1983.

[17] G.Cooper, Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks (Research note), Artificial Intelligence, 42:393-405, 1990.

[18] P. Dagum and M. Luby, On the approximation of Probabilistic Inference, Technical Report May, Section on Medical Informatics, Stanford University School of Medicine, Stanford, CA., 1994.

[19] Georg Wittenburg, A Survey of Extensions to Bayesian Networks, 2004.

[20] Daphne Koller and Avi Pfeffer, Object-Oriented Bayesian Networks, In Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97), pages 302-313, 1997.

[21] Y. Xiang, D. Poole, and M.P. Beddoes, Multiply-Sectioned Bayesian Networks and Junction Forests for Large Knowledge-Based Systems. Computational Intelligence, Volume 9, Issue 2, Pages 171-220, 1993.

[22] Y. Xiang, Comparison of Multiagent Inference Methods in Multiply Sectioned Bayesian Networks, International Journal of Approximate Reasoning 33 (2003) 235-254, 2002.

[23] Glenn Shafer, Probabilistic Expert Systems, The Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, PA., 1996.

[24] Prakash P. Shenoy, Binary Junction trees for Computing Marginals in the

Shenoy-Shafer Architecture, International Journal of Approximate Reasoning, 17(2-3):239-263, 1997.

[25] Dan Wu, On-demand Thrifty Propagation for Belief Updating in Bayesian Networks – A Preliminary Report, FLAIRS Conference 2005: 862-863, 2005.

[26] Pfeffer, A., Probabilistic Reasoning for Complex Systems, Ph. D. Thesis, Standford University, 2000.

[27] Pearl, J., Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, San Francisco, California: Morgan Kaufmann Publishers, 1988.

[28] Jensen, F. V., K. G. Lauritzen and K. G. Olesen, An Algebra of Bayesian Belief Universes for Knowledge-based Systems, Networks, 20(5), 637-659, 1990.

[29] Jensen, F. V., K. G. Lauritzen and K. G. Olesen, Bayesian Updating in Causal Probabilistic Networks by Local Computation, Computational Statistics Quarterly, 4, 269-282, 1990.

[30] Shenoy, P. P. and G. Shafer, Propagating Belief Functions Using Local Computation, IEEE Expert, 1(3), 43-52, 1986.

[31] Shenoy, P. P. and G. Shafer, Axioms for Probability and Belief-function Propagation, In R. D. Shachter, T. S. Levitt, J. F. Lemmer and L. N. Kanal (eds), Uncertainty in Artificial Intelligence, 4, 169-198, North-Holland, Amsterdam, 1990.

[32] Shenoy, P. P., Valuation-based Systems: A Framework for Managing Uncertainty in Expert Systems, In L. A. Zadeh and J. Kacprzyk (eds.), Fuzzy Logic for the Management of Uncertainty, 83-104, John Wiley & Sons, New York, NY., 1992.

[33] Cannings, C., E. A. Thompson and M. H. Skolnick, Probability Functions on Complex Pedigrees, Advances in Applied Probability, 10, 26-61, 1978.

[34] Lepar, V., Performance of Architecture for Local Computations in Bayesian Networks, PhD thesis, Institute of Informatics, University of Fribourg, Fribourg, Switzerland, in preparation, 1998.

[35] Almond, R., Graphical Belief Modeling. Chapman & Hall, London, 1995.

[36] Almond, R. and A. Kong, Optimality Issues in Constructing a Markov Tree from Graphical Models, Research Report No. 329, Department of Statistics, University of Chicago, Chicago, IL., 1991.

[37] S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen, HUGIN—A Shell for Building Bayesian Belief Universes for Expert Systems, In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence pages 1080-1085, 1989.

[38] F. Jensen, U. B. Kjærulff, M. Lang, and A. L. Madsen, HUGIN—The Tool for Bayesian Networks and Influence Diagrams, In First European Workshop on Probabilistic Graphical Models, pages 212-221, 2002.

[39] Cormen, T.; Leiserson, C.; and Rivest, R., Introduction to Algorithms. Cambridge, Massachusetts: The MIT press, 1997.

[40] C Language. http://en.wikipedia.org/wiki/C_programming_language.

[41] C.J. Butz, The Relational Database Theory of Bayesian Networks, Ph.D. Thesis, Departmentof Computer Science, University of Regina, Regina, Saskatchewan, 2000.

[42] S. K.M. Wong, D. Wu, and C.J. Butz, Triangulation of Bayesian Networks: a Relational Database Perspective, Lecture Notes In Computer Science, Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing, Pages: 389-396, 2002.

[43] D. Wu and S.K.M. Wong, Global Propagation in Bayesian Networks vs Semijoin Programs in Relational Databases, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 13(5): 539-560, 2005.

[44] Nevin L. Zhang, Computational properties of two exact algorithms for Bayesian networks, Applied Intelligence 9, 173-183, 1998.

[45] Saeed Ghahramani, Fundamentals of Probability, Prentice-Hall Inc., 2000.

[46] Richard L. Scheaffer, Introduction to Probability and Its Applications, Duxbury Press, 1994.

[47] Hugin API Reference Manual, Hugin Expert A/S, Version 6.2, 2004.

VITA AUCTORIS

Miss Liu He was born in 1982 in Sichuan Province, China.

She started studying Computer Science in Southwest Jiaotong University in 1999 and obtained her bachelor degree in Computer Science in 2003, and later worked as a network administrator.

In fall 2004, Miss He enrolled as master student at School of Computer Science, University of Windsor. She studied there for the next two years under Dr. Dan Wu's supervision.

Personal Information:
Name: Liu He
Email: hev@uwindsor.ca