2013

Testing and improving local adaptive importance sampling in LJF local-JT in multiply sectioned Bayesian networks

Sonia Bhatti

TESTING AND IMPROVING LOCAL ADAPTIVE IMPORTANCE SAMPLING IN LJF LOCAL-JT IN MULTIPLY SECTIONED BAYESIAN NETWORKS

By

SONIA BHATTI

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

Windsor, Ontario, Canada

2012

TESTING AND IMPROVING LOCAL ADAPTIVE IMPORTANCE SAMPLING IN LJF LOCAL-JT IN MULTIPLY SECTIONED BAYESIAN NETWORKS

by

SONIA BHATTI

APPROVED BY:

_____

Dr.Yunbi An, External Reader
Odette School of Business

_____

Dr. Yung H.Tsin, Internal Reader
School of Computer Science

_____

Dr. Dan Wu, Advisor
School of Computer Science

_____

Dr. Hongxuan(Karen) Jin, Co-Advisor
School of Computer Science

_____

Dr. Imran Ahmad, Chair of Defense
School of Computer Science

Defense Date: October 11, 2012

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Multiply Sectioned Bayesian Network (MSBN) provides a model for probabilistic reasoning in multi-agent systems. The exact inference is costly and difficult to be applied in the context of MSBNs. So the approximate inference is used as an alternative. Recently, for reasoning in MSBNs, LJF-based Local Adaptive Importance Sampler (LLAIS) has been developed for approximate reasoning in MSBNs. However, the prototype of LLAIS is tested on Alarm Network (37 nodes). But further testing on larger networks has not been reported. In this thesis, LLAIS algorithm is tested on three large networks namely Hailfinder (56 nodes), Win95pts (76 nodes) and PathFinder (109 nodes), to measure for its reliability and scalability. The experiments done show that LLAIS without parameters tuned shows good convergence for Hailfinder and Win95pts but not for Pathfinder network. However, when the parameters are tuned the algorithm shows considerable improvement in its accuracy for all the three networks tested.

Acknowledgment

First of all, I would like to thank my supervisor Dr. Dan Wu for advising and supporting me throughout the thesis with his patience and knowledge. I have learned a lot under his guidance, he always guided me in understanding the problem and giving me the way to find its solution. I was naïve when he took me under his supervision but he taught me how to set standards for myself so that I could improve myself. Dr. Karen H. Jin, Lecturer in Department of Computer Science, University of New Hampshire, USA, who guided me in spite of being outside the University of Windsor. She gave her valuable time and unselfish support throughout the thesis work.

I offer my sincere gratitude to Dr. Dan Wu and Dr. Karen H Jin, without them my thesis would never have been completed or written.

I would also like to thank my external reader, Dr. An, my internal reader, Dr. Tsin, and my thesis committee chair, Dr. Ahmad, for making time to be in my thesis committee, reading the thesis and providing valuable input. I appreciate all your suggestions, which have helped improve the quality of this thesis.

In my daily work I have been blessed with a friendly and cheerful group of fellow students, want to thank them for their friendship, kindness and moral support.

I would like to express my gratitude to my parents for their prayers and endless love, paying my tuition fee and giving me the opportunity to study in University of Windsor and in the end my thanks to the omnipresent God, for answering my prayers and giving me strength.

Table of Content

Declaration of Originality ..... **iii**
Abstract ..... **iv**
Acknowledgment ..... **v**
List of Figures ..... **viii**
List of Tables ..... **x**
1. Introduction **1**
1.1 Motivation 1
1.2 Objective 4
1.3 Overview of Thesis 6
1.4 Thesis Contribution 7
2. Background Study **9**
2.1 Concepts in Probability 9
2.1.1 Probability Theory 9
2.2 Probabilistic Graphical Models 11
2.3 Dependency Model 13
2.4 Bayesian Network - Definition and Example 14
2.5 Inference in Bayesian Networks 16
2.5.1 Exact Inference 17
2.5.2 Approximate Inference 20
2.6 Sampling in BN 23
2.6.1 Mathematical Foundation for Importance Sampling 25
2.6.2 Importance Sampling for BN 27
2.7 Multiply Sectioned Bayesian Network (MSBN) 30
2.8 Linked Junction Tree Forests (LJFs) in MSBNs 34
2.8.1 Initialization of LJF 36
2.9 Inference in MSBN 38
2.9.1 Importance Sampling for LJF local-JT 40
2.10 Discussion 44
3. Adaptive Importance Sampling in Bayesian Network **45**
3.1 Adaptive Importance Sampling in BN 45
3.1.1 Algorithm for AIS-BN 46
3.2 LJF Local Adaptive Importance Sampling 48
3.2.1 Algorithm for LLAIS 49
3.3 Discussion 50
4. Methods for Testing and Improving LLAIS **51**
4.1 Motivation 51
4.2 Network Selection 52
4.3 Experiments for Testing LLAIS 53
4.3.1 Performance Measures 53
4.3.2 Example for Testing Procedure 55
4.3.3 Design of Experiment 60
4.4 Improving LLAIS by Tuning the Tunable Parameters 61
4.5 Method for Testing the Improved LLAIS 65
4.5.1 Design of Experiment 65
4.5.2 Performance Measure 66
4.6 Discussion 66

5. Experiment Results for Testing and Improving LLAIS 68
5.1 Testing of LLAIS 68
5.2 Experiment Results 69
5.2.1 Testing Results on Hailfinder BN 70
5.2.2 Testing Results on Win95pts BN 77
5.2.3 Testing Results on Pathfinder BN 85
5.3 Summary of Testing 93
5.4 Tuning the Parameters and Improving LLAIS 94
5.5 Experiment Results 95
5.5.1 Experiment Results for Improved LLAIS on Hailfinder BN 95
5.5.2 Experiment Results for Improved LLAIS on Win95pts BN 97
5.5.3 Experiment Results for Improved LLAIS on Pathfinder BN 99
5.6 Summarizing the Experiments in terms of Time Taken 101
5.7 Discussion 104
6. Conclusion 105
6.1 Thesis Summary 105
6.2 Future Work 107
Bibliography 108
Vita Auctoris 113

List of Figures

Figure 2.1: The Asia Travel Network-simple BN 16

Figure 2.2: The Graph G in (a) is sectioned into G_0, G_1 and G_2 in (b) ψ in (c) is a hypertree over G 34

Figure 2.3: (a) A BN (b) A small MSBN with three subnets (c) the corresponding MSBN hypertree 36

Figure 2.4: An MSBN LJF shown with initial potentials assigned to all the three subnets.. ..... ….37

Figure 2.5: Inter-agent message passing 39

Figure 2.6: Sampling local JT with {abc} as root 43

Figure 4.1: Procedure for testing LLAIS 59

Figure 5.1: Steps for testing LLAIS on larger networks 69

Figure 5.2: Structure of Hailfinder Network with 56 nodes and 66 edges 70

Figure 5.3: Performance comparison of approximate importance function and exact importance function on Hailfinder network with 9 evidence nodes 71

Figure 5.4: Performance comparison of approximate importance function and exact importance function on Hailfinder network with 11evidence nodes 72

Figure 5.5: Performance comparison of approximate importance function and exact importance function on Hailfinder network with 13 evidence nodes 73

Figure 5.6: Frequency distribution of P(E) in Hailfinder network ……...74

Figure 5.7: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Hellinger's distance* for Hailfinder network 75

Figure 5.8: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Mean Square Error* for Hailfinder network. 77

Figure 5.9: Structure of Win95pts network with 76 nodes and 112 edges 78

Figure 5.10: Performance comparison of approximate importance function and exact importance function on Win95pts network with 9 evidence nodes 79

Figure 5.11: Performance comparison of approximate importance function and exact importance function on Win95pts network with 11 evidence nodes 80

Figure 5.12: Performance comparison of approximate importance function and exact importance function on Win95pts network with 13 evidence nodes 81

Figure 5.13: Frequency distribution of P(E) in Win95pts network 81

Figure 5.14: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Hellinger's distance* for Win95pts network 83

Figure 5.15: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Mean Square Error* for Win95pts network 84

Figure 5.16: Structure of Pathfinder network with 109 nodes 85

Figure 5.17: Performance comparison of approximate importance function and exact importance function on Pathfinder network with 9 evidence nodes 86

Figure 5.18: Performance comparison of approximate importance function and exact importance function on Pathfinder network with 11 evidence nodes 87

Figure 5.19: Performance comparison of approximate importance function and exact importance function on Pathfinder network with 13 evidence nodes 88

Figure 5.20: Frequency distribution of P(E) in Pathfinder network 89

Figure 5.21: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Hellinger's distance* for Pathfinder network 91

Figure 5.22: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Mean Square Error* for Pathfinder network 92

Figure 5.23: Performance comparison of original LLAIS and improved LLAIS for Hailfinder network.*Hellinger's distance* for each of the 30 test cases cases plotted against P(E) 96

Figure 5.24: Performance comparison of original LLAIS and improved LLAIS for Win95pts network. *Hellinger's distance* for each of the 30 test cases cases plotted against P(E) 98

Figure 5.25: Performance comparison of original LLAIS and improved LLAIS for Pathfinder network. *Hellinger's distance* for each of the 30 test cases cases plotted against P(E) .. 100

List of Tables

Table 4.1: Shows the comparison of values of various tunable parameters for Original LLAIS and Improved LLAIS 65

Table 5.1: Statistical Results for all the 30 test cases generated to test LLAIS for Hailfinder network in terms of *Hellinger's distance* 75

Table 5.2: Statistical Results for all the 30 test cases generated to test LLAIS for Hailfinder network in terms of *Mean Square Error* 76

Table 5.3: Statistical Results for all the 30 test cases generated to test LLAIS for Win95pts network in terms of *Hellinger's distance* 82

Table 5.4: Statistical Results for all the 30 test cases generated to test LLAIS for Win95pts network in terms of *Mean Square Error* 84

Table 5.5: Statistical Results for all the 30 test cases generated to test LLAIS for Pathfinder network in terms of *Hellinger's distance* 90

Table 5.6: Statistical Results for all the 30 test cases generated to test LLAIS for Pathfinder network in terms of *Mean Square Error* 91

Table 5.7: Statistical Results for all the 30 test cases generated to compare the performance of Original LLAIS with Improved LLAIS on Hailfinder network in terms of *Hellinger's distance* 96

Table 5.8: Statistical Results for all the 30 test cases generated to compare the performance of Original LLAIS with Improved LLAIS on Win95pts network in terms of *Hellinger's distance* 98

Table 5.9: Statistical Results for all the 30 test cases generated to compare the performance of Original LLAIS with Improved LLAIS on Pathfinder network in terms of *Hellinger's distance* 100

Table 5.10: Comparison of the average time taken for ten test cases by approximate importance function and exact importance function in LLAIS in producing the posterior probabilities 102

Table 5.11: Comparison of the average time taken for ten test cases by Original LLAIS and Improved LLAIS in producing the posterior probabilities 103

Chapter 1

Introduction

1.1 Motivation

The Bayesian network is a directed acyclic graph (DAG) representing the probabilistic relationship in a complex system. The Bayesian network model has been used over the last 25 years as a tool for managing uncertainty using probability. It is basically used to represent knowledge. As the computational power of Bayesian network is increasing day by day so it is used as an effective tool to explore and explain complex problems. In the last few years a lot of techniques have been developed in order to assess and solve belief networks, various belief networks have been available today and are used by many diagnostic reasoning systems, for example, MUNIN, ALARM, Pathfinder and QMR-DT.

In [1], the intelligent agent or computational system is the one that can sense the surrounding and then take necessary actions according to the set targets, these agents process local observations produce required decisions and later on make execution of the chosen decisions and take actions. A probabilistic agent uses probabilistic knowledge representations and reasons with regard to the state of the domain. In recent years the systems involving the multiple agents have become quite prevalent. In the multi-agent paradigm, a set of cooperative agents make use of their local knowledge and inter-agent communication to collectively reason about the state of uncertain domain. For instance we can think of problem [1] in which four driverless cars on city streets will cooperate

with each other and coordinate in their actions so as to avoid any accident and safely pass a four-way-stop intersection. So, the main challenge faced today is the adequate utilization and extension of the current representation models and available inference algorithms for the single agent paradigm to multi-agent settings.

Multiply Sectioned Bayesian Network (MSBN) is the model grounded on the idea of cooperative multi-agent probabilistic reasoning. It is an extension of the traditional Bayesian network model and provides us with solution to the probabilistic reasoning under cooperative agents. From [2], these agents working in cooperation are assigned many different tasks depending on the type of application; one of the common tasks is to make estimation about the true state of the domain so that they can act accordingly. An MSBN consist of a set of inter-related Bayesian subnets and each subnet encodes agent's knowledge on sub domain. In order to make multi-agent inference, the existing methods for inference in single-agent Bayesian network (BN) have been extended. The Multiple agents [1] collectively and cooperatively reason about the problem domain on the basis of their local knowledge, local observation and limited inter-agent communication.

Many existing inference calculations in MSBN are generally carried out in some secondary structure which is known as *linked junction tree forest* (LJF). An LJF constitutes local junction trees (JT) and linkage trees to make connections between the neighboring agents.

It has been seen that message passing in Hugin-based architecture is quite expensive and

also it is impractical to carry out the efficient calculation in case of MSBNs due to excessive computational time and memory requirements. Though many efforts have resulted to be advantageous in developing the approximate techniques for Bayesian networks but still a lot of research has to be done in extending these solutions to MSBNs. As discussed in [2], the probabilistic inference in MSBN is performed in distributed fashion. The algorithms for multi-agent inference in MSBNs are the extension of methods for inference in single-agent Bayesian network, for example message passing in junction trees.

The important problem to approach is the issue of feasibility of probabilistic inference when the size of practical models available today is increasing in size from few variables to several hundreds of variables. The exact inference has been proved to be NP-hard [3], so the approximate inference techniques are used to estimate the posterior probabilities. The approximate algorithms belong to the family of stochastic sampling algorithms which is also called stochastic simulation or Monte-Carlo algorithms. It is very important to study the practicability and convergence properties of sampling algorithms on large Bayesian networks.

Localized stochastic sampling:

To date there are many stochastic sampling algorithms proposed for Bayesian networks and are widely used in BN approximation but this area is taken to be quite problematic. Many attempts have been made in developing MSBN approximation algorithms but all of these forgo the LJF structure and sample MSBN directly in global context. Also it has been shown that such type of approximation requires more inter-agent message passing

and also leaks the privacy of local subnet [4]. Hence, sampling MSBN in global context is not good idea as it analyzes only small part of the entire multi-agent domain space. So in order to examine local approximation and to maintain LJF framework, the sampling process is to be done at each agent's subnet. The *LJF-based local adaptive importance sampler* (LLAIS) is an example of the extension of BN importance sampling techniques to JT's. An important aspect of this algorithm is that it facilitates inter-agent message calculation along with the approximation of the posterior probabilities.

1.2 Objective

Since the exact inference is considered to be expensive and difficult as the problem domain becomes larger and complex, so the approximate inference algorithms are being developed. The algorithm LLAIS is used for approximate reasoning in LJF local-JT in MSBN. It is the application of adaptive importance sampling on LJF local-JT to produce the posterior probabilities of local beliefs. The prototype of LLAIS has been tested on smaller network consisting of 37 nodes (Alarm network). In MSBN, the size of local JTs or subnets can vary so it is important to test the scalability and reliability of the algorithm when the size of local JT goes beyond 37 nodes. One way of testing the efficiency and reliability of approximate algorithms is to use them on the larger network.

The networks used for testing LLAIS are as follows:

(i) Hailfinder(56nodes) (ii) Win95pts(76nodes) and (iii) Pathfinder (109nodes).

Each network represents in itself the size of local JT in MSBN. Hence we limited ourselves up to 109 nodes network; this size is quite agreeable to test the algorithm since

it is local adaptive importance sampling which is applied locally on the subnets or local-JTs and therefore, to deal with this much big size of local JT will be quite appropriate as the local JTs are formed after the sectioning of the large BN.

For experimentation, these networks are taken from the Genie and Smile [5]. The testing of LLAIS will include comparing its sampling output (using approximate importance function) with that from using exact importance function which is considered to be the optimal one.

The comparison of performance using the exact importance function will help in knowing how close the approximate importance function in LLAIS is able to reach the optimal results. It is believed that computing the exact importance function will also affect the running time of algorithm since it is the optimal and do not require updating and learning of importance function as required by approximate importance function and hence saving a lot of time.

Further, there are various tunable parameters in LLAIS that will be discussed in chapter 4, we believe if the values of these tunable parameters are tuned properly it may lead to the improvement in algorithm in terms of time efficiency and accuracy. Hence to summarize the objective of this thesis is firstly, to test LLAIS algorithm for its scalability and reliability by applying it on larger networks and secondly, tuning the various tunable parameters to improve the accuracy and time efficiency of the algorithm.

1.3 Overview of Thesis

The outline of the thesis is as follows:

- **Chapter 2: Background Study** – In this chapter we will be giving a brief introduction to the probability theory. Various concepts and notations will be concisely discussed giving readers idea about the background of the probability theory. Section 2.2 discusses the probabilistic graphical models and why they are important to study. We will also be giving gentle introduction to the Bayesian networks and inference in Bayesian network including their mathematical and technical concepts. We will discuss major exact and approximate inference techniques; focussing more on the approximate inference in BN and hence discussing it in detail. This chapter will talk about the multi-agent reasoning with MSBN and Linked Junction Forests (LJFs) and how inference is done in LJF. Basically we tried to cover as much as possible the literature review of the graphical models, BNs, MSBNs and reasoning in BN and MSBN, giving more emphasis on the approximate algorithms for inference.

- **Chapter 3: Adaptive Importance Sampling in Bayesian Networks –** This chapter discusses the adaptive importance sampling applied in the context of Bayesian networks and linked junction forests in MSBN. In this chapter we will be talking about the algorithm LLAIS which is to be tested and explaining it in detail.

- **Chapter 4: Methods for testing and Improving LLAIS –** This chapter discusses the methods and experiments procedure followed in testing the scalability and reliability of LLAIS. Further the improvement in LLAIS algorithm is also discussed by comparing its performance with the original LLAIS algorithm.

- **Chapter 5: Testing and Improving LLAIS with Experiment Results –** This chapter discusses the experiment results of the testing of LLAIS on the larger networks. The comparison of the results of LLAIS improved with original LLAIS will also be discussed. On the whole this chapter include the graphs plotted for the experiments results along with tables showing the information of comparisons made.

- **Chapter 6: Conclusion –** This chapter includes the summarization of the thesis with some directions for future research.

1.4 Thesis Contribution

As discussed so far the application of LLAIS is done on smaller network consisting of 37 nodes which is treated as local JT in LJF. LLAIS produced good estimates of local posterior beliefs for this smaller network but its further application on larger networks is not reported. So in this thesis, we tested LLAIS for its scalability and reliability on the three larger networks treating them as local JTs in MSBN. It is important to test the algorithm since the size of local JT can vary and go beyond 37 nodes network, on which preliminary testing has been done. Our testing demonstrated that LLAIS is quite scalable for the 56 and 76 nodes network but once it is applied to 109 nodes network its performance deteriorates. The calculation of the exact importance function resulted in saving a lot of time since it does not need updating and learning as required by the approximate importance function, hence making the algorithm quite time efficient. Further, since there are various tunable parameters in LLAIS when these parameters are

tuned properly it results in significant improvement in the performance of algorithm; the improved LLAIS requires less number of samples and less updates than required by the original algorithm to give better results.

Chapter 2

Background Study

In this chapter, a brief introduction to the probabilistic graphs will be discussed, in particular about the Bayesian network and Multiply Sectioned Bayesian network. The inference including exact and approximate inference techniques will also be discussed.

2.1 Concepts in Probability

2.1.1 Probability Theory

The probability is the study of uncertainty. One of the most common notions of probability theory is random variable. A random variable is a variable whose values are outcome of a particular experiment. Just as the other variables random variables can take any different values. From [1], all the possible outcomes of random variables are mutually exclusive and collectively exhaustive. These outcomes together as a set form the *domain* of the variables. The probability of a random variable is measured by a function that maps each possible outcome, or instantiation, of this random variable into the interval [0, 1].

Notations: Capital letters such as A, B or X_i denote random variables. Bold capital letters, such as **X** or **Y**, denote sets of variables and **E** usually denote the set of evidence variables. Lower case letters, such as *a* and *x* denote particular instantiation of variable **A** and **X** respectively. Bold lower case letters, such as **x** and **y** denote particular instantiation

of sets **X** and **Y** respectively while the bold lower case letter **e** is used to denote the observation for the set of variables **E.**

Given the set of random variables as $V = \{V_1, V_2 \ldots\ldots, V_n\}$, joint probability is defined as a probabilities of all combinations of the possible outcome of each variable in V.

Joint probability distribution or JPD is denoted as:

$$P(V) = P(V_1 = v_1, V_2 = v_2, \ldots\ldots\ldots\ldots, V_n = v_n)$$

$$= P(v_1, v_2, \ldots\ldots\ldots\ldots, v_n),$$

where $v_1, v_2, \ldots\ldots\ldots\ldots, v_n$ are the respective values which those variables may take.

The domain of V is the cross join of the domains of all variables in $\{V_1, V_2 \ldots\ldots, V_n\}$; further, each element from the domain of a set of variables is known as an instantiation of these variables.

The Marginalization is defined as the process of summing out some variables from the probability distribution. For example we can obtain the probability distribution of a subset X of V by summing out all the variables in a set of V excluding X (which is denoted as $V \backslash X$).

Hence the Marginal Probability Distribution (MPD) of X from $P(V)$ is denoted as :

$$P(X) = \sum_{V \backslash X} P(V)$$

where $P(X)$ is called the marginal probability distribution and can also be written as $P^{\downarrow x}(V)$.

It is to be noted that the probability distribution of some random variable is updated once it observes the realization of another random variable or in other words we can say that the probability distribution of random variables changes after receiving the information that another variable has taken up some value. This is the relation of dependency and is expressed as conditional probability distribution (CPD). Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ are two disjoint subsets of $\boldsymbol{V}$, and let $\boldsymbol{x}$ and $\boldsymbol{y}$ be their instantiations (or values).

Then the CPD of $\boldsymbol{X} = x$ given $\boldsymbol{Y} = y$ denoted as $P(\boldsymbol{X} = x|\boldsymbol{Y} = y)$ and also abbreviated as $P(x|y)$ is formulated as:

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{P(\boldsymbol{x},\boldsymbol{y})}{P(\boldsymbol{y})} \tag{2.1}$$

where $\mathrm{P}(\boldsymbol{y}) \neq 0, also\ \boldsymbol{X}\ is\ the\ head\ and\ \boldsymbol{Y}\ is\ the\ tail\ of\ this\ CPD.$

The conditional probability distribution of some variable $\boldsymbol{X}$ with given evidence **e,** is denoted as $P(X|\boldsymbol{E} = \boldsymbol{e})$ is also known as *posterior probability distribution* of $\boldsymbol{X}$.

2.2 Probabilistic Graphical Models

The probabilistic graphical models provide ground to reason for uncertainties in real world applications. These models use the knowledge given to them to make conclusions. They play a key role in modeling uncertainties in the real world.

From [6] for example, sometimes a doctor might have to take information about the patient- his name, symptoms, test results, personal characteristics to reach to the conclusion what disease he might be suffering and what course of treatment has to be followed, but in complex systems there are many uncertainties, since real world is

affected by many factors, it is because due to large systems we are often not sure about the true state of the system. It may be due to the fact that our observation is partial or it may be due to that only some aspect of the world is being exposed to us. As a result, it might happen that true disease of the patient is not observed directly or even the future prognosis made by him is never observed. Further it has to be kept in mind that due to lack of observation we are not clear about the true state of world, so we can say that relationships are not deterministic; hence it can happen that there are very few diseases where we can have true relationship between the disease and its symptoms and even fewer such relationships between the disease and its prognosis. So there was a need for reasoning system to take into account all the different possibilities about the state of world.

The probability theory provides us with the formal framework where multiple possible outcomes and their likelihood can be considered. So the probabilistic framework helped in deterministic specification of the behavior of the complex system

The probabilistic graphical models are significant tool in helping the agent to reason about its uncertain domain and taking the action accordingly. These models use graphical representation to represent the complex probabilistic distribution. The graphical models are described [7] as representation of probabilistic structure along with functions that are used to derive the joint distribution. From [1], the probabilistic graphical models merge together the representation and algorithmic power of both the probability theory and the

graph theory where the data is modeled as a set of nodes which represent random variables and the connecting arcs represent the dependencies between the variables.

2.3 Dependency Model

The probabilistic model for a set of random variables is defined by joint probability distribution but in order to specify a probability model using full JPD is an impractical task. Since the domain described by n boolean variables requires a table of size $O(2^n)$ and takes $O(2^n)$ time to process that table, so to lower down this cost we have to take into account the advantages of dependence and independence relationship among variables.

Let $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$ be disjoint subsets of $\boldsymbol{V}$.

$\boldsymbol{X}$ and $\boldsymbol{Y}$ are unconditionally independent if the following conditions will hold:

$$P(\boldsymbol{X}|\boldsymbol{Y}) = P(\boldsymbol{X}),\ P(\boldsymbol{Y}) \neq 0. \tag{2.2}$$

The above unconditional independency can be denoted as conditional independency statement (CIS) $I(\boldsymbol{X}, \Phi, \boldsymbol{Y})$ or $I(\boldsymbol{X}, \boldsymbol{Y}|\Phi)$**.**

$\boldsymbol{X}$ and $\boldsymbol{Y}$ are conditionally independent given $\boldsymbol{Z}$ if the following holds:

$$P(\boldsymbol{X}|\boldsymbol{Y}, \boldsymbol{Z}) = P(\boldsymbol{X}|\boldsymbol{Z}),\ P(\boldsymbol{Z}) \neq 0. \tag{2.3}$$

The above conditional independency relation can be expressed as *conditional independency statement (CIS)* ***I(X, Z, Y)*** or ***I(X, Y|Z).***

We can conclude that dependency model is any model M of a set of variables V denoted as $V = \{V_1, V_2, \ldots\ldots\ldots, V_n\}$, from where we can decide whether ***I(X,Y|Z)*** is true or not for all possible disjoint $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$.

An easy and direct way to model dependency models is to use directed acyclic graphs (DAG). DAG stands for directed acyclic graph consist of set of nodes as the random variables, and a set of directed links between nodes but with no directed cycles. The Bayesian network is represented in the form of DAG. To identify the independency relationship in DAG, concept called d-separation is used.

Definition 2.1: *D-separation*

Let G be a Directed Acyclic Graph and X, Y, Z be disjoint set of nodes in G. A path ρ between nodes $x \in X$ and $y \in Y$ is closed by Z if one of the following two conditions holds: (1) There exists $z \in Z$ that is either tail-to-tail or head-to-tail on ρ. There exists a node v that is head-to-head on ρ and neither v or any descendant of v is in Z. If both conditions fail, then ρ is rendered open by Z.

Nodes x and y are d-separated by Z if every path between x and y is closed by Z;

X and Y are d-separated by Z if for every $x \in X$ and $y \in Y$, x and y are d-separated by Z.

2.4 Bayesian Network - Definition and Example

The Bayesian networks have been seen as a powerful tool in Artificial Intelligence in order to simulate and approximate real situations. They are defined as probabilistic graphical model for reasoning under uncertainty [1]. It provides the coherent framework for the various decision support systems which function using uncertain knowledge available to them, such as in machine learning, bioinformatics, medical diagnosis and so on.

In [7], Bayesian networks also known as belief networks, causal probabilistic networks, directed Markov fields or influence diagrams (given some additional structure). It is a directed acyclic graph in which nodes represent random variables, and arcs represent direct probabilistic dependence between directly connecting variables. Each node is assigned probabilistic distribution conditioned on that node's parents. Then Joint probability distribution determined from the factorization of CPD's assigned.

From [8], **Bayesian Network** is a triplet $B = (V, G, P)$. V is a set of variables, G is a connected DAG whose nodes correspond to one-to-one to members of V such that each variable is conditionally independent of its non-descendants given its parents. Each variable V_i in V is represented as a node in DAG, it is associated with CPD denoted by P which is defined as:

$$P = \{P(V_i|Pa(V_i))|V_i \in V\}$$

Here the $Pa(V_i)$ denote the parents of node V_i in the DAG. The product of these CPD's defines JPD given as:

$$P(V) = \prod_{V_i \in V} P\left(V_i \middle| Pa(V_i)\right), \quad (2.4)$$

Equation 2.4 defines the factorization which is also known as Bayesian factorization done in terms of CPDs. The DAG G is commonly referred to as dependency structure of Bayesian network.

Hence we conclude that Bayesian network models by providing the compact representation of JPD also captures the independency among random variable.

Consider the simple BN in the Figure 2.1 named Asia travel network.

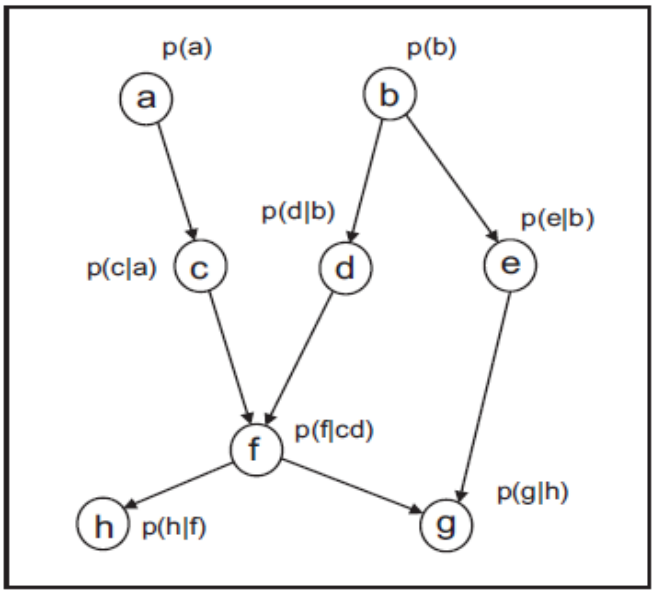

Figure 2.1: The Asia Travel Network-simple BN.

The Asia Travel Bayesian network is DAG defined over the set V of variables where $V = \{a, b, c, d, e, f, g, h\}$,

the corresponding set of CPDs is:

$$P = \{p(a), p(b), p(c|a), p(d|b), p(e|b), p(f|cd), p(h|f), p(g|ef)\},$$

These two components will now define Joint Probability distribution (JPD) and expressed as:

$$P(V) = p(a).p(b).p(c|a).p(d|b).p(e|b).p(f|cd).p(h|f).p(g|ef), \qquad (2.5)$$

the CPD factorization in above Equation 2.5 gives the Bayesian factorization of $P(V)$.

2.5 Inference in Bayesian Networks

The assignment of values to the observed variables is known as evidence. The most important form of reasoning used in Bayesian networks is called belief updating which involves computing the posterior probability distribution of the variables of interest when

the value of their evidence is given. A lot of research is being done in the field of reasoning. In [9], the Bayesian network is known to provide the efficient and natural way for modeling the causal structures along with the computational basis for probabilistic inference. The inference task in the Bayesian network is represented as $P(\boldsymbol{X}|\boldsymbol{E}=e)$ where $\boldsymbol{X}$ denotes the set of variables and $\boldsymbol{E}$ denotes evidence set; while set of variables $\boldsymbol{H}$ is denoted as hidden variables and is represented by $\boldsymbol{H}=\boldsymbol{V}\backslash\boldsymbol{X}\backslash\boldsymbol{E}$. Since we have joint probability distribution defined over the random variables so probabilistic inference can be performed by summing out the hidden variables using the sequence of multiply and addition operations.

2.5.1 Exact Inference

The calculation of exact value of posterior probability is called exact inference.

The following Equation 2.6 calculates for the exact value for inference according to the probability theory [1]:

$$P(\boldsymbol{X}|\boldsymbol{E}=e)=\frac{P(\boldsymbol{X},e)}{P(e)}=\alpha P(\boldsymbol{X},e)=\alpha\sum_{H}P(\boldsymbol{X},e,\boldsymbol{H}), \tag{2.6}$$

where α is a normalization value $1/P(e)$.

So, in order to calculate the value of $P(\boldsymbol{E}=e)$, we need to perform summation over all variables except for the evidence variables in $P(\boldsymbol{X}\backslash\boldsymbol{E},\boldsymbol{E}=e)$. There are different ways to process summation giving rise to different algorithms, only idea behind them is getting the value of evidence using minimum number of computations. It is a combinatorial optimization problem [7]. The Enumeration algorithm for computing the posterior probability is not feasible when applied to large networks, due to number of arithmetic

operations involved. So, many have been developed to make the inference approachable and easy.

As asserted in [10], In 1980s, Pearl gave the algorithm for efficient **message passing for inference** for polytrees but it was regarded as polynomial time complexity in number of nodes, later he gave exact inference algorithm for multiply connected networks called **loop cutset conditioning**. The loop cutset conditioning algorithm works by changing the connectivity of the multiply connected graph rendering it singly connected graph and instantiating the selected subset of nodes that are referred as loop cutset. The complexity for this algorithm is calculated from the number of different instantiations that need to be considered and resulting in its time complexity growing exponentially with the size of loop cutset being $O(dc)$, where d is the number of values which random variables can take and c giving the size of loop cutset so here it is required to reduce the size of loop cutset in case of multiply connected networks but the problem of finding the minimum loop cutset is known to be NP-hard.

The **variable elimination (VE) algorithm** works by eliminating variables other than the queries one by one by summing out them. The complexity of VE can be measured by the number of numerical multiplications and numerical summations it performs. An *optimal elimination ordering* is one that results in the least complexity. The problem of finding an optimal elimination ordering is NP-complete.

In [10] the most popular exact inference algorithm is junction tree propagation developed

by Lauritzen and Spiegelhalter [11]. The inference task is performed on secondary structure of BN which is junction tree. A junction tree (JT) is a graph which is formed by the nodes that are subsets of the domain variables called clusters or cliques.

The following steps will elaborate in detail about the conversion of Bayesian network to the junction tree:

Step 1. Moralizing the original graph: Moral graph is formed by connecting the pair of nodes that have common child, that is, two nodes with same child are said to be married, and then replacing the directed edges with undirected edges.

Step 2. Triangulating the moral graph: In a triangulated graph, for every cycle of length greater than or equal to four, a link is drawn between two non-adjacent nodes on the cycle. The problem for finding the optimal triangulation is NP-complete [12], but fast triangulation algorithms that can produce high quality results are available [13] [14].

Step 3. Identifying the cliques: After the triangulation has been performed we identify the cliques. A clique or cluster is nothing but a maximal complete subgraph. Every clique corresponds to the node of junction tree.

The JT is constructed using an important property which is called running intersection property [1] which says that if a variable belong to two distinct JT clusters, then it should belong to every cluster on the path connecting the two clusters. So taking this property as the basis the set of common nodes to a pair of neighboring clusters are defined as their *Separators*. The construction of optimal JT is discussed in [15]. Once the junction tree is formed, every clique is assigned initial function ∅ which is called potential of the clique.

It is product of all the conditional probability distributions which the clique has received. If a clique has not received any CPD it will be initialized to 1. The initial potential of the cluster does not represent cluster marginal so message passing has to be done so that the information of probability distribution of each cluster is made consistent to the other clusters.

The Hugin architecture [16] [17] and the Shenoy-Shafer architecture [18][19][20] are the two major variations for the JT-based exact inference calculations. The clique tree propagation works well with sparse networks but its performance gets affected as the size of network increases. Its complexity is exponential to the size of the largest clique made out from the undirected graph.

But unfortunately the problem of exact inference in Bayesian networks is NP-hard. So as the researchers were faced with intractability of exact inference in large and complex networks, so it led them to investigate to develop the approximate inference algorithms as an alternative.

2.5.2 Approximate Inference

Today when the size of practical models is increasing to the size of hundreds of variables, the problem of finding the feasibility of probabilistic inference becomes important. The exact inference algorithms including the JT algorithm become impractical and [7] when applied to larger and complex networks it require either prohibitive amount of memory or a prohibitive amount of computation and unable to complete.

Although the approximation algorithm to the desired precision is also shown to be NP-hard [21] but they are the only alternative which can produce any result at all.

As discussed in [1] [7], the various approximate inference techniques developed so far are:

Model Simplification: This method simplify the original structure of model in some way and hence weakening the network dependencies and then exact methods can be applied to the simplified network so as to obtain the approximation solution. These simplification methods involve the reduction in the cardinality of the size of JT clusters [22], some methods reduce the model complexity by annihilating small probabilities [10], Sarkar's algorithm approximates the Bayesian network by finding the optimal tree-decomposable representation which is closest to the actual network. Another most widely used method is reducing the edges of an original network. Some simplification methods also involve using the variational methods for fitting parameters to simple logistic function [23] [10].

Search Based Algorithms: Search based methods assumes that small fraction of joint probability mass contain the majority of probability mass. It finds the high probability instantiations with large probability mass in the joint probability distribution and uses them to obtain reasonable approximations. These methods give good approximations of the network with almost all extreme conditional probabilities. These include Henrion's "Top-N" search based methods [24], Poole's search approach using conflicts [25] [26] [27].

Loopy Belief Propagation: As discussed in [28], there has been a lot of research going on the use of Pearl's polytree propagation in Bayesian network with loops. In [29] researchers have analytically demonstrated that loopy belief algorithm can perform quite well in error-correcting codes and computer vision. It performs well on the graphs with loops but fail to give good convergence when the density of graph increases resulting in poor results.

Stochastic Sampling Methods: The stochastic sampling algorithms also known as Monte-Carlo algorithms are the most well- known and most commonly used simulation methods. These algorithms generate randomly selected instantiations of the network as per the probabilistic distributions of the model and then the frequencies of these instantiations are calculated for nodes of interest as an approximation of the inference. The accuracy of these algorithms depends upon the size of samples irrespective of the topology of the network. The most important characteristic of the stochastic sampling algorithm is its nice any-time property such that the computation can be interrupted at any given time in order to yield an approximation [10].

The main idea behind the stochastic sampling algorithm which is the class of algorithm under approximate inference is that these algorithms sample the probability distribution and estimate the probability of queries depending upon the samples obtained by calculating the frequencies of instantiations of interest. The advantage of using the stochastic sampling is that the execution time is independent of the topology of network and is linear with number of samples. Also these algorithms have any real-time property

such that their computation can be interrupted at any time with guaranteed results [7].

2.6 Sampling in BN

The sampling algorithms are the most common approximate inference technique used for calculating the posterior probabilities. The main idea behind the sampling is to randomly instantiate each node in Bayesian network in topological order and produce single sample. These samples are generated a number of times and finally after some specific number of times or after generating some specific amount of samples the posterior probabilities are calculated for each node by counting the frequency of each possible instantiation of every node in all the samples generated. Now we will discuss some of the common sampling algorithms:

1. **Forward Sampling:** It is the simplest sampling algorithm. In this we start by ordering the nodes in topological order, assign the values of evidence variables and number of samples generated. The sampling will proceed by first sampling the parent node and then the child node. The evidence nodes are instantiated to observed state and so omitted from sample generation. The root node is randomly instantiated to one of its possible states as per the prior probability of this node and child node is instantiated depending upon the parent node instantiation, to one of its possible states according to the conditional probability distribution. The procedure is followed a number of times, once n samples are generated, the posterior probability $P(e)$ is obtained calculating the ratio between the total score sum and the number of samples.

2. **Likelihood Weighting:** The LW sampling is same to the forward sampling but it never discards the sample [43] [44]. It assigns weight to each sample generated. Suppose we have Bayesian Network which has evidence nodes as $E_1, E_2, \ldots, E_n$ that are instantiated to $e_1, e_2, \ldots, e_n$ respectively. Then the weight of the sample s will be calculated as $P(E_1 = e_1 | Pa(e_1)) * P(E_2 = e_2 | Pa(e_2)) *, \ldots\ldots\ldots, * P(E_n = e_n | Pa(e_n))$. In other words the weight of the sample is the product of the probability that each evidence node will have the desired value given the value of its parents in the sample s. Once n number of samples are generated the posterior probability is calculated for each node X and each possible value x by summing out weight of sample in which X is instantiated to x and divided by total weight of all n samples.

It is most commonly used simulation method for Bayesian network inference. It is simple and is able to increase the precision by generating more number of samples than the other algorithms in the same amount of time. Its convergence deteriorates when the evidence is very unlikely.

3. **Importance Sampling:** Importance sampling is same as the generic sampling algorithm [44]. The importance sampling has two variants called *self-importance* (SIS) and *heuristic importance*. Here the importance function is updated trying to revise the conditional probability tables periodically in order to make the sampling distribution gradually approach the posterior distribution. From [45], since the data used to update the importance function and to compute the estimator, this process introduces bias in the estimator. There are certain algorithms that are combination of *self-importance* and

heuristic importance [44] [46] but promising direction in the work on sampling algorithms has not been achieved yet. Now, we will be going through the theoretical roots of importance sampling since it is the basic step to understand in order to learn about the existing know-how of stochastic sampling algorithms for Bayesian networks which will ground the basis of our thesis.

2.6.1 Mathematical Foundation of Importance Sampling

Let $g(X)$ be the function of m variables $\boldsymbol{X} = (X_1, \ldots\ldots, X_m)$ over a domain $\Omega \subset R^m$ such that computing $g(X)$ for any $\boldsymbol{X}$ is feasible. Consider the problem of approximate computation of the integral

$$I = \int_{\Omega} g(\boldsymbol{X}) \boldsymbol{dX} \tag{2.7}$$

Importance sampling approaches this problem by writing the integral (2.7) as

$$I = \int_{\Omega} \frac{g(\boldsymbol{X})}{f(\boldsymbol{X})} f(\boldsymbol{X}) \, \mathrm{d}\mathbf{X} \tag{2.8}$$

where $f(\boldsymbol{X})$ often called importance function, is a probability density function over Ω. The samples can be generated from $f(\boldsymbol{X})$ using importance sampling if the importance function is zero only when the original function is zero, that is, $g(\boldsymbol{X}) \neq 0 \Rightarrow f(\boldsymbol{X}) \neq 0$.

Once the samples have been generated from *n* points as $s_1, s_2, \ldots\ldots, s_n \in \Omega$, according to the probability density function $f(\boldsymbol{X})$, we can estimate the integral *I* by

$$\hat{I}_n = \frac{1}{n} \sum_{i=1}^{n} \frac{g(s_i)}{f(s_i)} \tag{2.9}$$

And estimate the variance of $\widehat{I_n}$ by:

$$\hat{\sigma}^2\left(\hat{I}_n\right) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \left(\frac{g(s_i)}{f(s_i)} - \hat{I}_n\right)^2 \quad (2.10)$$

The estimator thus obtained has the following properties:

1. $E\left(\hat{I}_n\right) = I$

2. $\lim_{n\to\infty} \hat{I}_n = I$

3. $\sqrt{n}.\left(\hat{I}_n - I\right) \xrightarrow{n\to\infty} Normal\left(0, \sigma^2_{f(X)}\right), where$

$$\sigma^2_{f(X)} = \int_{\Omega} \left(\frac{g(\boldsymbol{X})}{f(\boldsymbol{X})} - I\right)^2 f(\boldsymbol{X}) \mathrm{d}\mathbf{X}$$

4. $E\left(\hat{\sigma}^2\left(\hat{I}_n\right)\right) = \hat{\sigma}^2\left(\hat{I}_n\right) = \frac{\sigma^2_{f(X)}}{n}$

The variance $\hat{I}_n$ is proportional to $\sigma^2_{f(X)}$ and inversely proportional to the number of samples. To minimize the variance we can either increase the number of samples or we can reduce $\sigma^2_{f(X)}$. Taking into consideration latter, [47] reports the following theorem and corollary:

Theorem 2.6.1: The minimum of $\sigma^2_{f(X)}$ is equal to

$$\sigma^2_{f(X)} = \left(\int_{\Omega} |g(\boldsymbol{X})| \mathrm{d}\mathbf{X}\right)^2 - I^2$$

And occurs when X is distributed according to the following probability density function

$$f(\boldsymbol{X}) = \frac{|g(\boldsymbol{X})|}{\int_{\Omega} |g(\boldsymbol{X})| \mathrm{d}\mathbf{X}}$$

Corollary 2.6.1: If $g(\boldsymbol{X}) > 0$, then the optimal probability density function is

$$f(\boldsymbol{X}) = \frac{g(\boldsymbol{X})}{I}$$

And $\sigma^2_{f(X)} = 0$.

In practice, sampling precisely from $f(\boldsymbol{X}) = \frac{g(\boldsymbol{X})}{I}$ will happen very rarely but we still expect that the functions close to it can help in reducing the variance effectively. Usually the closer the shape of the function $f(\boldsymbol{X})$ is to the shape of the function $g(\boldsymbol{X})$, the smaller is $\sigma^2_{f(X)}$. It is essential to put in more strength towards choosing the importance function whose shape is as close as possible to that of $g(\boldsymbol{X})$ than to apply the Brute-force method of increasing the number of samples.

It is worth noting that when $f(\boldsymbol{X})$ is uniform then the importance sampling becomes general Monte-Caro sampling. One another property of importance sampling is that one should avoid $f(\boldsymbol{X}) \ll |g(\boldsymbol{X}) - I \cdot f(\boldsymbol{X})|$ in any domain of sampling even if $f(\boldsymbol{X})$ matches well with $g(\boldsymbol{X})/I$, it is because in this case the variance can become very large or even infinite. In order to adjust this we can make $f(\boldsymbol{X})$ to be larger in unimportant regions of the domain of $\boldsymbol{X}$.

Till now we discussed about the importance sampling for continuous variables but the results discussed remains valid for discrete variables where the integration is substituted with summation.

2.6.2 Importance Sampling for BN

From [1], the family of stochastic sampling belongs to the BN approximate algorithm class. The importance sampling is commonly used simulation technique which is used to sample modified distribution called importance function. The underlying idea is to

approximate the average over a set of numbers by an average over a set of sampled numbers.

In order to evaluate the sum $I = \sum_{x \in X} g(x)$ for some real function g, samples are generated from the importance function f such that $g(x) \neq 0 \Rightarrow f(x) \neq 0$, we have

$$I = \sum_{x \in X} g(x) = \sum_{x \in X} \frac{g(x)}{f(x)} f(x) = E_f \left[\frac{g(x)}{f(x)} \right] \tag{2.11}$$

By the definition of expected value we can estimate I as

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} w(x^i), \tag{2.12}$$

where $w(x^i) = \frac{g(x^i)}{f(x^i)}$, is called sample weight or score.

In order to compute the probability of evidence $P(\boldsymbol{E} = e)$ from a JPD $P(\boldsymbol{X}) = \prod_{i=1}^{n} P(X_i | Pa(X_i))$ of a BN model, we have to sum over all the non-evidence nodes:

$$P(\boldsymbol{E} = e) = \sum_{\mathbf{X} \setminus \mathbf{E}} P(\mathbf{X} \setminus \mathbf{E}, \mathbf{E}) \tag{2.13}$$

$$= \sum_{\boldsymbol{X} \setminus \boldsymbol{E}} \prod_{i=1}^{n} P(X_i | Pa(X_i), \mathbf{E} = \mathbf{e})$$

Let $\boldsymbol{Z} = \boldsymbol{X} \setminus \boldsymbol{E}$, we simplify the above equation 2.13 as:

$$P(\mathbf{E} = \mathbf{e}) = \sum_{z \in \boldsymbol{Z}} P(Z = z, \boldsymbol{E} = \boldsymbol{e}) \tag{2.14}$$

We can apply the principal of importance sampling.

Assume that our proposal distribution or sampling distribution Q is the importance function such that $P(Z = z, \boldsymbol{E} = \boldsymbol{e}) \neq 0 \Rightarrow Q(Z = z) \neq 0$.

Equation 2.14 can be re-written as:

$$P(\mathrm{E} = \mathbf{e}) = \sum_{\boldsymbol{z} \in \boldsymbol{Z}} \frac{P(Z=z, \mathbf{E}=\mathbf{e})}{Q(Z=z)} Q(Z = z) \tag{2.15}$$

By the definition of expected value, we have

$$E_Q|\boldsymbol{Z}| = \sum_{\boldsymbol{z}\in\boldsymbol{Z}} z\, Q(Z = z)$$

Equation 2.15 becomes:

$$P(\mathbf{E} = \mathbf{e}) = E_Q\left[\frac{P(Z=z,E=e)}{Q(Z=z)}\right] = E_Q[w(Z = z)] \qquad (2.16)$$

where $w(Z = z)$ denote the score of each sample and is calculated as:

$$w(Z = z) = \frac{P(Z = z, \boldsymbol{E} = \boldsymbol{e})}{Q(Z = z)}$$

Suppose we sampled from Q and obtained a sample set $(z_1, z_2, \ldots\ldots\ldots, z_n)$, then

$$\hat{P}(\boldsymbol{E} = \boldsymbol{e}) = \frac{1}{N}\sum_{i=1}^{N}\frac{P(Z=z_i,E=e)}{Q(Z=z_i)} = \frac{1}{N}\sum_{i=1}^{N} w(Z = z_i) \qquad (2.17)$$

As the size of sample increases the expected value will approach the true average. It means we can say that as $N \to \infty, \hat{P}(\boldsymbol{E} = \boldsymbol{e}) = P(\boldsymbol{E} = \boldsymbol{e})$, thus such estimator is unbiased.

In order to obtain the posterior probability $P(\boldsymbol{X}|\boldsymbol{E})$ we can separately compute the two terms $P(\boldsymbol{X}, \boldsymbol{E})$ and $P(\boldsymbol{E})$, and then combine them by the definition of conditional probability.

$$\hat{P}(X_i = x_i|\boldsymbol{E} = \boldsymbol{e}) = \frac{\hat{P}(X_i=x_i,\boldsymbol{E}=\boldsymbol{e})}{\hat{P}(\boldsymbol{E}=\boldsymbol{e})} = \frac{\sum_{j=1}^{N}\delta(x_i,z_j)w(z_j)}{\sum_{j=1}^{N}w(z_j)} \qquad (2.18)$$

where $\delta(x_i, z_j) = 1$ if and only if the sample z_j contains $X_i = x_i$, Otherwise $\delta(x_i, z_j) = 0$.

It is important to note here that two terms $P(\boldsymbol{E} = \boldsymbol{e})$ and $P(X_i = x_i|\boldsymbol{E} = \boldsymbol{e})$ can be separately estimated unbiasedly, the estimation thus obtained by combining them through Equation 2.18 is not an unbiased estimator [45].

The quality of importance function depends upon how close the sampling distribution is to the true distribution. Many importance sampling algorithms have been developed so far for Bayesian networks where choice of importance function may vary from the prior distribution as in the likelihood weighting algorithm [6] to more refined choices such as there exist algorithms that update the importance function through learning processes [45] or calculate the importance function directly with loopy belief propagation [48].

The main aim of these methods is to ultimately reach the optimal importance function, which is a function proportional to the posterior distribution and should have a thick tail [49] [50]. This section has discussed in detail about the mathematical foundation of importance sampling and how it is applied in context of Bayesian networks for approximate reasoning.

2.7 Multiply Sectioned Bayesian Network (MSBN)

Today intelligent systems are being applied to the larger and complex domains and there are many applications that are found to be suitably addressed by multi-agent systems [30] [31]. In [6], multi-agent system is the one which consist of a number of agents interacting with each other typically by the exchange of messages through the computer network

infrastructure. In general cases, the agents will be acting or representing users or owners having different goals and motivations.

The problem domain in multi-agent systems is distributed naturally among the agents and typically with increased size and complexity, so in order to model such a domain as single BN becomes difficult and performing inference becomes challenging [8]. As a result it is natural to consider one single, large and complex domain being divided into subdomains; where each subdomain is individually represented and managed by a relatively light weighted single agent. The basic assumption taken is that these agents are expected to be cooperative in the sense that they will always provide truthful information about their local domains to other agents.

The Multiply Sectioned Bayesian Network (MSBN) [8] extends the traditional BN model from a single agent oriented paradigm to the distributed multi-agent paradigm and provides a framework to apply probabilistic inference in distributed multi-agent systems. Under MSBNs, a large domain can be modeled modularly and the inference task can be performed in coherent and distributed fashion.

The **MSBN model** is based on the following five assumptions:

1. Agent's belief is represented as probability.
2. Agents communicate their beliefs based on a small set of shared variables.
3. A simpler agent organization is preferred.
4. A DAG is used to structure each agent's knowledge.

5. An agent's local JPD admits the agent's belief of its local variables and the shared variables with other agents.

An **MSBN** [32] consist of inter-related Bayesian subnets and each subnet encodes agent's knowledge on a subdomain. The agents are organized in *hypertree* structure and exchange of messages is done through *hyperlink* between the adjacent agents. The complexity of communication among all the agents is linear on the number of agents and the complexity of local inference is the same as if subnet is a single agent based Bayesian network.

The MSBN is described in terms of the following definitions [8].

Definition 2.2: *Let $G = (V, E)$ be a connected graph, with the set of random variables V and connecting edges E, sectioned into subgraphs $\{G_i = (V_i, E_i)\}$.*

Let the subgraphs be organized into an undirected tree ψ where each node is uniquely labeled by a G_i and each link between G_k and G_m is labeled by the non-empty interface $V_k \cap V_m$ such that for each i and j, $V_i \cap V_j$ is contained in each subgraph on the path between V_i and V_j in ψ. Then ψ is a hypertree over G. Each G_i is a hypernode and each interface is a hyperlink.

Definition 2.3: *Let G be a directed graph such that a hypertree over G exists. A node x contained in more than one subgraph with its parents $Pa(x)$ in G is a d-sepnode if there exists at least one subgraph that contains $Pa(x)$. An interface I is a d-sepset if every $x \in I$ is a d-sepnode.*

Definition 2.4: *A hypertree MSDAG* $G = \cup_i G_i$ *where each* G_i *is a DAG, is a connected DAG such that (1) there exists a hypertree* ψ *over G, and (2) each hyperlink in* ψ *is a d-sepset.*

Definition 2.5: *An MSBN M is a triplet (V, G,P).* $V = \cup_i V_i$ *is the domain where each* V_i *is a set of variables.* $G = \cup_i G_i$ *(a hypertree MSDAG) is the structure in which nodes of each DAG* G_i *are labeled by elements of* V_i*. Let* x *be a variable and* $Pa(x)$ *be all the parents of* x *in G. For each* x*, exactly one of its occurrences (in a* G_i *containing* $(\{x\} \cup Pa\{x\})$ *is assigned* $P(x|Pa(x)),$ *and each occurrence in other DAGs is assigned a uniform potential.* $\mathrm{P} = \prod_i P_i$ *is the JPD, where each* P_i *is the product of the potentials associated with nodes in* G_i*. A triplet* $N_i = (V_i, G_i, P_i)$ *is called a subnet of M. Two subnets* N_i *and* N_j *are said to be adjacent if* G_i *and* G_j *are adjacent on the hypertree MSDAG.*

Figure 2.2 below shows an example [2] where subnets in an MSBN are satisfying the hyper tree condition. Here $V_2 \cap V_1 = \emptyset$ (hence the hypertree condition is trivially satisfied). But in general $V_i \cap V_j$ can be non-empty. The interface between the subnets in an MSBN must form a d-sepset. So here each of the a, b, c, j, k in the interfaces is a d-sepnodes. Hence, the interfaces $\{a, b, c\}$ and $\{j, k\}$ are d-sepsets. If we reverse the arcs from j to l, the node j would no longer be a d-sepnode consequently $\{j, k\}$ would no longer be a d-sepset. Both the hypertree and d-sepset conditions ensure syntactically that the agents can communicate their belief by passing messages over the interfaces only.

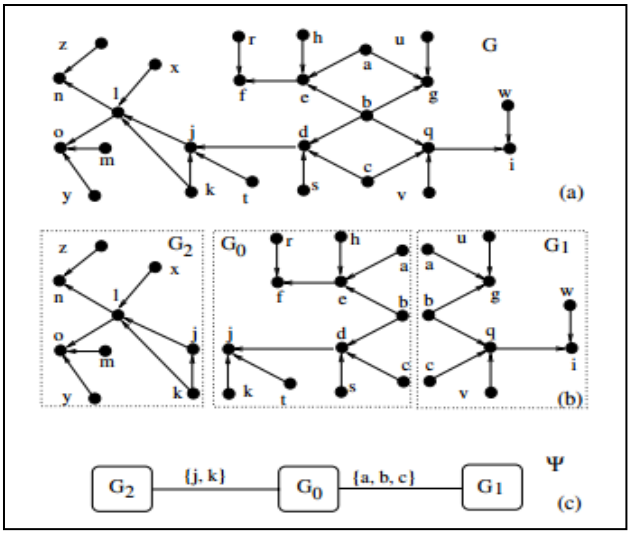

Figure 2.2: The Graph G in (a) is sectioned into G_0, G_1 and G_2 in (b). ψ in (c) is a hypertree over G.

An MSBN consists of a set of inter-related Bayesian subnets, each of the subnets encoding the agent's knowledge on the subdomain. Every agent maintains its local BN subnet which represents the partial view of the entire larger problem domain. The union of all subnet DAGs must be a DAG and these subnets are organized into hypertree structure. Each node in hypertree corresponds to the subnet and each hyperlink corresponds to d-sepset, which are the shared variables between the adjacent subnets. The hyperlink renders the two sides of the network conditionally independent. MSBN provides a framework for uncertainty reasoning in cooperative multi-agent systems. Today MSBN have been used in many fields such as building surveillance [33], medical and equipment diagnosis [34][35]. MSBN have also been used to provide support for the object-oriented Bayesian networks [36].

2.8 Linked Junction Tree Forests (LJFs) in MSBNs

The inference or belief updating in MSBN is usually compiled into a secondary structure

called *linked junction tree forest* (LJF). We can say that LJF is derived dependence structure which is adopted for distributed probabilistic inference in MSBNs. An LJF [37] is constructed through the process of cooperative and distributed compilation so that each *hypernode* in *hypertree* is transformed into a *local JT* and each *hyperlink* into a *linkage tree*. A linkage tree is nothing but it is a special name given to the junction tree constructed from a d-sepset. In a linkage tree, each cluster is called a linkage and each separator is known as linkage separator. The cluster in the local JT which contains a linkage is called Linkage host. The two adjacent subnets maintain their own linkage tree corresponding to the same d-sepset.

The Linked Junction Forest LJF as described in [8]:

Definition 2.6: *A linked Junction Forest is a tuple (V,G,T,L)*

$V = \cup_i V_i$ *is the total universe where each* V_i *is a set of variables called a subdomain,*

$G = \cup_i G_i$, *where each* $G_i = (V_i, E_i)$ *is a chordal graph such that there exist a hypertree* ψ *over G.*

$T = \{T_i\}$ *is a set of JTs, each of which is a corresponding JT of* G_i.

$L = \{L_i\}$ *is a collection of linkage tree sets. Each* $L_i = \{L_{i,j}\}$ *is a set of linkage trees, one for each hyperlink incident to* G_i *in* ψ. *Each* $L_{i,j}$ *is a linkage tree of* T_i *with respect to a hyperlink* $V_i \cap V_j$.

Figure 2.3 shows the Bayesian network which is sectioned into three subnets and its corresponding MSBN hypertree structure. The three subnets being formed are G_1, G_2 and G_0 and are maintained by agents A_1, A_2 and A_0 respectively. Once the hyper

tree structure is formed, it is converted to the LJF and inference is carried out in this LJF structure which is secondary structure of MSBN. Figure 2.4 shows the LJF constructed from the MSBN in Figure 2.3. Here the local Junction trees are T_1, T_2 and T_0 are constructed from the BN subnets G_1, G_2 and G_0 respectively, shown in the solid line boxes. The linkage trees are derived from the d-sepsets and enclosed in dotted line boxes. Each pair of adjacent subnets maintains identical linkage trees. For example, the linkage tree L_{02} contains the linkage abc and bcd and their linkage hosts in T_0 are the clusters $\{abc\}$ and $\{bcd\}$.

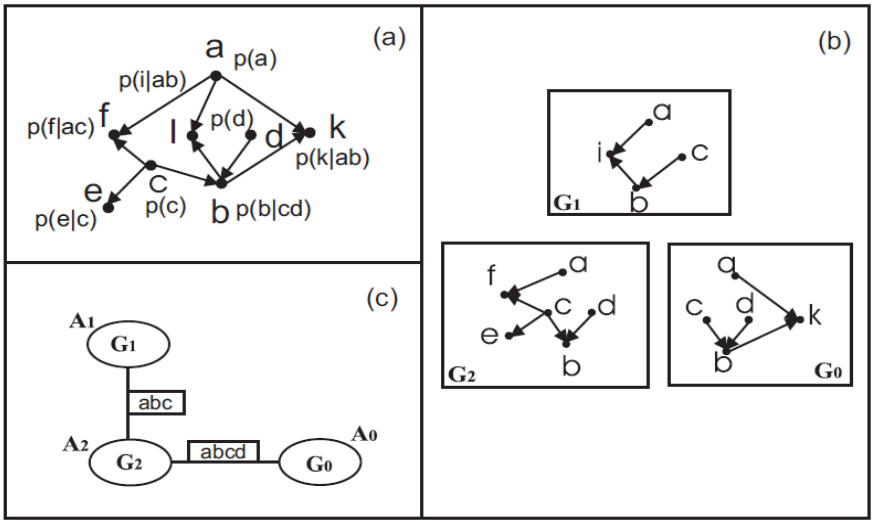

Figure 2.3: (a) A BN (b) A small MSBN with three subnets (c) the corresponding MSBN hypertree.

2.8.1 Initialization of LJF

During the initialization process of LJF, [37] exactly one of all the occurrences of a variable x (from the subnet containing $\{x\} \cup Pa\{x\}$) is assigned the CPD $P(x|Pa(x))$ and all other occurrences are assigned unity potential. Along with it unity potential is also assigned to the separators in the local JT and linkages in the linkage tree of LJF. Figure 2.4 shows the initialization process where from all the seven occurrences of the variable

b, only one occurrence of b in cluster $\{bcd\}$ of local junction tree T_2 is assigned the CPD $p(b|cd)$ and rest, all other occurrences are assigned unity potential.

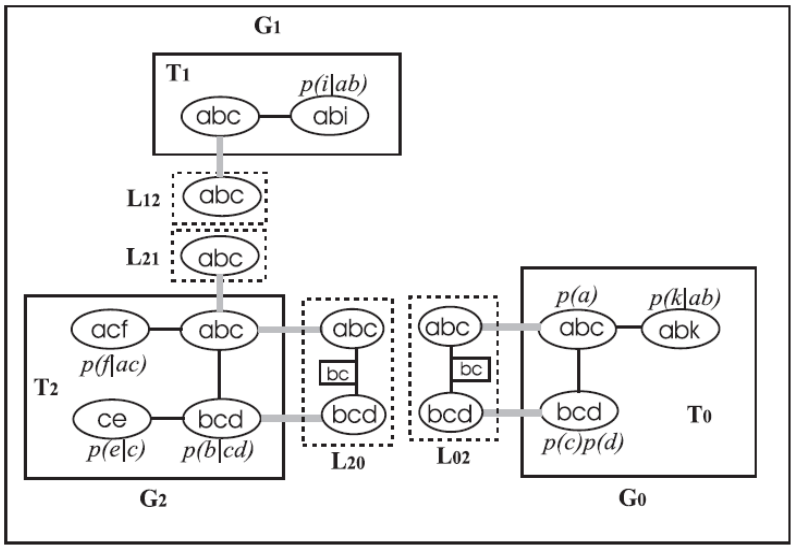

Figure 2.4: An MSBN LJF shown with initial potentials assigned to all the three subnets.

The initial potential of a local JT is either the product of all of its assigned CPDs or 1 if no CPD is assigned. This initial potential in general does not provide complete information for an agent to correctly reason about its own problem subdomain. The reason behind it is that local JT is yet to be consistent and more important thing is that the potential of each subnet does not represent the JPD of its local variables called prior marginal distribution. The initial potentials of all the three subnets in Figure 2.4 can be represented as: $\emptyset(G_0) = P(a).P(c).P(d).P(k|ab)$, $\emptyset(G_1) = P(i|ab)$ and $\emptyset(G_2) = P(b|cd).P(e|c).P(f|ac)$ but none of these potentials forms the JPD over the corresponding local variables. Although it is possible to achieve local consistency through message passing in the local JT, inter-agent communication is necessary to provide each MSBN subnet the missing information to form the prior marginal.

This process is known as marginal calibration.

2.9 Inference in MSBNs

The belief updating in case of MSBN is done in its secondary structure which is called Linked Junction Tree Forest (LJF). As discussed in the previous section the LJF consist of local junction tree each of which is constructed from an MSBN subnet. The communication between the adjacent subnets is done through the linkages which act as interface between two adjacent local JTs. The main concept behind a linkage tree is that it renders the two sides of MSBN network conditionally independent. The agents reason about its subdomains within the local JTs and then collaborate to solve distributed inference problems by communicating over the linkage trees [38].

Many algorithms have been proposed extending from BN message passing schema to the existing MSBN LJF inter-agent message passing for exact belief updating [39][40]. The belief propagation process in MSBN requires two rounds of global message passing on the corresponding LJF. This process can be related similar to the message passing in JTs but here only difference is we are using it in the contexts of MSBN subnets. The messages are passed recursively inward and outward relatively to the pre-selected root node. During the inward message passing each agent passes message to its neighbor towards the root's direction and during the outward message passing each agent passes a message to its neighbor away from the root's direction beginning with the root node itself. Hence in total two messages will be passed over each single linkage.

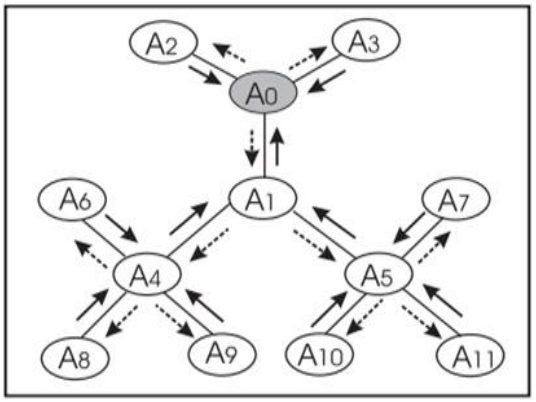

Figure 2.5: Inter-agent message passing.

For example, in Figure 2.5 let the shaded node A_0 represents as root node. The solid arrows indicate the direction of inward message starting from the all the leaf nodes while dashed arrows indicate the direction of outward message passing originating from the root A_0.

From [41], the first general inference method in MSBNs [42], [40] is extension of the junction tree based inference method [16] for single-agent BN's. In [41], they have called this method as product based inference with linked junction forest (LJF), it allows for the efficient exact inference when done autonomously in a cooperative multi agent system when the dependence structure is dispersed. The inference performed by an agent is said to be autonomous if it can performed by an agent alone without communicating with other agents, and later after the inference has been done, the agent is able to answer all the probabilistic queries exactly conditioned on the local knowledge and observations as well as on the global knowledge and observations till the last communication.

However it has been seen that [38] performing exact inference in context of MSBN is very expensive process. There are two reasons behind this: First, the local belief of each subnet has to be represented exactly similar to Bayesian network. Secondly, the local belief of each subnet needs to be recomputed every time whenever a new inter agent message arrives. For example, in the above Figure 2.5, the root node A_0 has to compute its local potential three times for the messages it will receive from A_1, A_2 and A_3. The amount of re-calculation done in each of the subnet is not dependent upon the selection of root but to the network topology. For the large networks except for the simpler ones such repeated update of the exact belief requires prohibitive amount of resources. Hence agents have to perform repeated message passing in order to maintain local consistency during the message calculation.

2.9.1 Importance Sampling for LJF local-JT

The research done so far has highlighted many difficulties in applying stochastic sampling to MSBNs at global level [2]. Along with it even the direct local sampling in MSBN subnet is not possible since these subnets lack characteristics for valid BN structure. So in this regard the LJF local JT, which is secondary structure of a subnet, can be calibrated using marginal over all the local variables [37] and hence making local sampling possible. Many algorithms have been proposed to be applied on calibrated LJF local JT which combine sampling with the JT belief propagation [6] [51] [52]. Many of the algorithms developed are based on the Markov Chain Monte Carlo and thus do not support efficient inter-agent message calculation in case of MSBNs [1].

In [4], the concept of JT-based importance Sampler is introduced, although the importance sampling in JT was previously done by [53] but [4] introduced explicit form of importance function which will facilitate the learning of optimal sampling distribution as well as provide for the efficient inter-agent message calculation.

From [4], the JPD over all the variables in a calibrated local JT can be obtained similar to Bayesian network DAG factorization. Let $C_1, C_2 \ldots\ldots, C_m$ be the m JT clusters given in the ordering which satisfies the running intersection property. The separator $S_i = \emptyset$ for $i = 1$ and $S_i = C_i \cap (C_1 \cup C_2 \cup \ldots\ldots \cup C_{i-1})$ for $i = 2,3, \ldots, m$. Since $S_i \subset C_i$, the residuals are defined as $R_i = C_i \backslash S_i$. The junction tree running intersection property guarantees that the separator S_i separates the residual R_i from the set $(C_1 \cup C_2 \cup \ldots\ldots \cup C_{i-1}) \backslash S_i$ in junction tree.

Applying chain rule to partition the residues given by the separators and have JPD expressed as:

$$P(C_1, \ldots\ldots, C_m) = \prod_{i=1}^{m} P(R_i|S_i) \tag{2.19}$$

The main idea is to select the root from the JT clusters and then directing all the separators away from the root forming a directed sampling JT. It is analogous to BN since both follow recursive form of factorization.

Once the JPD has been defined for LJF local JT, the importance function P' is defined as follows:

$$P'(\boldsymbol{X} \backslash \boldsymbol{E}) = \prod_{i=1}^{m} P(R_i \backslash \mathbf{E}|S_i)|_{\mathbf{E}=\mathbf{e}} \tag{2.20}$$

The vertical bar in $P(R_i \backslash \mathbf{E}|S_i)|_{\mathbf{E}=\mathbf{e}}$ indicates the substitution of **e** for **E** in $P(R_i \backslash \mathbf{E}|S_i)$.

The importance function in Equation 2.20 is factored into a set of local components each corresponding to the JT clusters. It means that when the calibrated potential is given on each JT cluster C_i we can easily compute for every cluster the value of $P(R_i|S_i)$ directly. For the root cluster:

$$P(R_i|S_i) = P(R_i) = P(C_i), i = 0. \tag{2.21}$$

The sampling JT is traversed and sampling is done on the variables of residue set in each cluster corresponding to the local conditional distribution. This sampling is taken to be similar to the BN sampling except now group of nodes are being sampled and not the individual nodes. Whenever cluster is encountered with the node in the evidence set **E**, it will be assigned value which is given by evidence assignment. A complete sample consist of the assignment to all the non- evidence nodes according to the local JT's prior distribution.

The score for each sample can be computed as:

$$Score_i = \frac{P(s_i, \boldsymbol{E})}{P\prime(s_i)} \tag{2.22}$$

The score so computed in Equation 2.22 will be used in the LLAIS algorithm for adaptive importance sampling as we will see in the next chapter 3. Consider the example shown in Figure 2.6 generating the samples for the local JT T_0 with cluster abc as the root cluster. Suppose the evidence is observed at g, because sampling has to be performed in

topological order so first sampling is done over variables abc since it is root cluster according to its local importance function.

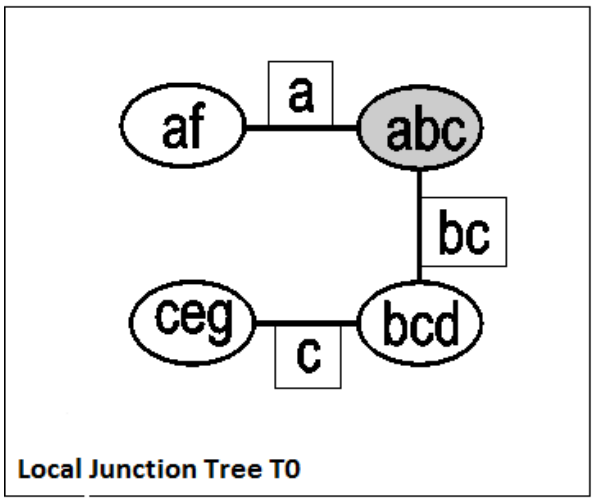

Figure 2.6: Sampling local JT with {abc} as root.

Now, for the cluster *af*, only f will be sampled from its local importance function since a is already sampled. For the cluster bcd, only d will be sampled according to the local importance function given the already sampled values of b and c and next we have to sample over e for cluster ceg given the values of g and c already determined.

Since it has been proven that the optimal distribution function for BN importance sampling is the posterior distribution $P(\mathbf{X}|\mathbf{E}=\mathbf{e})$[45], so applying the same rule to JTs, we can define the optimal importance function as:

$$\rho(\boldsymbol{X}\backslash\boldsymbol{E}) = \prod_{i=1}^{m} P(R_i\backslash\boldsymbol{E}|\boldsymbol{E}=\boldsymbol{e}) \qquad (2.23)$$

The above Equation 2.23 takes into account the influence of all the evidences from all clusters in the sample of current cluster.

2.10 Discussion

In this chapter, we have given a background study where it is emphasized that BNs have emerged as a powerful tool for reasoning under uncertainties. Although the exact inference in BN is proved to be NP-hard; hence many practical approximate algorithms are being developed to solve wide range of inference tasks. We discussed about approximate reasoning in detail by focusing on the various sampling techniques and mathematical background for importance sampling.

The extension of Bayesian networks to Multiply Sectioned Bayesian networks is also discussed in detail. MSBN provides with the model through which large and distributed domain can be modeled in the form of organized subdomains. Since MSBN maintain the hypertree structure so the BN JT inference algorithm can be easily and naturally extended to MSBN's secondary structure called LJF. We also discussed about the application of importance sampling to LJF local JT and how it helps in reasoning in case of MSBNs.

Despite the similarities between the BN JT and MSBN hypertree till date global inference in LJF has not been made effective. Since the exact inference in MSBN has been proved to be very costly so later in the thesis local inference in MSBN hypertree will be discussed through approximate methods for reasoning.

Chapter 3

Adaptive Importance Sampling in Bayesian Networks

In this chapter, adaptive importance sampling will be discussed in the context of Bayesian networks and how this sampling technique has led to the significant improvement in probabilistic inference. The adaptive importance sampling shows significant results where the stochastic sampling algorithms so far developed show poor convergence and also takes too long to converge to the reasonable estimates of the posterior probabilities as the probability of evidence goes more and more unlikely.

3.1 Adaptive Importance Sampling in BN (AIS-BN)

Since the existing stochastic algorithms convergence rate was quite slow in case of unlikely evidence. These algorithms fail to learn good importance function and also they fail to reduce the sampling variance. In 2000, Adaptive Importance Sampling algorithm (AIS-BN) was proposed by [45] that proved to be a significant achievement in field of approximate reasoning in BN by showing promising results in convergence rate even in case of unlikely evidence.

The importance function used by them is:

$$\rho(\mathbf{X}\backslash\mathbf{E}) = \prod_{i=1}^{n} P(X_i|Pa(X_i), \mathbf{E}) \quad (3.1)$$

The concept of **Importance Conditional Probability Tables (ICPT)** was introduced where ICPT of a node X is table of posterior probabilities $\mathrm{P}(\mathrm{X}|\mathrm{Pa}(\mathrm{X}), \mathbf{E} = \mathbf{e})$ conditional

on the evidence and indexed by its immediate predecessors, $Pa(X)$. These are the probability tables which are initially equal to the CPTs and then updated after each updating interval.

In [45], the learning rate is defined as the rate at which optimal importance function will be learned as per the formula $\eta(k) = a(\frac{a}{b})^{k/kmax}$, where a = initial learning rate, b = learning rate in the last step, k = number of updates, and $kmax$ = total number of updates.

The updating interval defines the number of samples after which AIS-BN will update the importance function.

Algorithm 3.1.1: AIS-BN

Step 1. Ordering the nodes in the topological order.

Step 2. Specify the total number of updates K, Desired number of samples M, Updating interval L.

Step 3. Initialize the importance function $Pr^0(X|E)$ same as $Pr(X|E) = \prod_{X_i \in X \setminus E} Pr\left(X_i \middle| Pa(X_i)\right) | E = e$, modifying it by applying heuristic cut off to handle small probabilities and changing the CPT tables of the parents of an evidence node E to uniform distribution only when $Pr(E = e) < 1/(2.n_E)$

Step 4. Generate L samples according to the current ICPT table. Update the importance function $Pr'(x_i|pa(X_i), e)$ based on the total samples.

Step 5. Update the ICPT tables based on the following learning function

$P^{K+1}(x_i|pa(X_i), \boldsymbol{e}) = Pr^k(x_i|pa(X_i), e) + \eta(k).(Pr'(x_i|pa(X_i), e) - Pr^k(x_i|pa(X_i), e))$,

where $\eta(k)$ is the learning rate.

Step 6. Modify the importance function with the heuristic of Є-cutoff. For the next update, goto Step 4. Update the importance function till $kmax$.

Step 7. Generating M samples and calculating the score arrays.

Step 8. Normalizing the score arrays for every node and getting the posterior probabilities.

There were certain heuristic initializations done in this algorithm for performance improvement: They are as follows:-

(i) two heuristics for initialization of importance function greatly affected the speed of convergence. (a)The importance function is taken to prior probability distribution over the network variables, P(X). (b) The ICPT tables of the parents of the evidence nodes to uniform distributions,

(ii) learning method for importance function,

(iii) dynamic weighting function for combining samples from different stages of the algorithm.

In [45], they stopped learning after 10 iterations to save time. The heuristics used have shown to accelerate the speed of learning process. Hence AIS-BN results in getting fairly good estimates of posterior probabilities in limited time and has been proven to show dramatic improvement in the convergence rates in case of large Bayesian networks when compared to other existing approximate inference techniques.

3.2 LJF local Adaptive Importance Sampling (LLAIS)

We discussed in chapter 2 about the importance sampling application to LJFs in MSBN, Here in this section we will be talking about the adaptive importance sampling applied in the context of local JT in MSBN.

In 2010, LJF local JT importance sampler called LLAIS [4] was designed that follows the principle of adaptive importance sampling as discussed in the section 4.1 for learning factors of importance function. This algorithm was specifically designed for the approximation of posteriors in case of local JT in MSBN providing the framework for calculation of inter-agent messages between the adjacent local JTs.

The sub-optimal importance function used for LJF local adaptive importance sampling is as follows,

$$\rho(\mathbf{X}\backslash\mathbf{E}) = \prod_{i=1}^{m} P(R_i\backslash\mathbf{E}|S_i, \mathbf{E} = \mathbf{e}) \tag{3.2}$$

This importance function is represented in the form of set of local tables. This importance function is learned to approach the optimal sampling distribution. These local tables are called the ***Clustered Importance Conditional Probability Table*** **(CICPT).** These CICPT tables are created for each local JT cluster consisting of the probabilities indexed by the separator to the precedent cluster (based on the cluster ordering in the sampling tree) and conditioned by the evidence. For non-root JT clusters, CICPT table are defined in the form of $P(R_i|S_i, \mathbf{E})$, and for the JT root cluster, CICPT table are of the form of $P(R_i|S_i, \mathbf{E}) = \mathrm{P}(C_i|\mathbf{E})$.

The learning strategy is to learn these CICPT tables on the basis of most recent batch of samples and hence the influence of all evidences is counted through the current sample set. These CICPT tables have the structure similar to the factored importance function and are alike to an ICPT table of Adaptive Importance Sampling of BN in the previous section 4.1 and are updated periodically by the scores of samples generated from the previous tables.

Algorithm 3.2.1: LLAIS

Step 1. Specify the total number of samples M, total updates K and update interval L, Initialize the CICPT tables as in Equation 3.2.

Step 2. Generate L samples with the scores according to the current CICPT tables. Estimate $P'(R_i|S_i, \boldsymbol{e})$ by normalizing the scores for each residue set given the states of separator set.

Step 3. Update the CICPT tables based on the following learning function [45]:

$$P^{K+1}(R_i|S_i, \boldsymbol{e}) = \big(1 - \eta(k)\big)P^k(R_i|S_i, \boldsymbol{e}) + \eta(k)P'(R_i|S_i, \boldsymbol{e}),$$

where $\eta(k)$ is the learning rate.

Step 4. Modify the importance function if necessary, with the heuristic of Є-cutoff. For the next update, goto Step 2.

Step 5.Generate the M samples from the learned importance function and calculate scores as in Equation 2.22 (From Chapter 2).

Step 6. Output the posterior distribution for each node.

In LLAIS the importance function is dynamically tuned from the initial prior distribution and samples obtained from the current importance function are used to refine gradually the sampling distribution. It is well known that thick tails are desirable for importance sampling in BNs. The reason behind it is that the quality of approximation deteriorates in the presence of zero probabilities due to generation of large number of samples having zero weights [15][109][32]. This issue is solved using the heuristic Є-cutoff [14], the small probabilities are replaced with Є if less than a threshold Є, and the change is compensated by subtracting the difference from the largest probability.

3.3 Discussion

In this chapter, we discussed about the application of adaptive importance sampling in case of full BN and MSBN subnets. AIS-BN is the adaptive importance sampling applied to singly connected Bayesian networks while the extension of it, LLAIS is applied to the local JT in MSBN. There is difference in the initialization of importance function between the two algorithms but both of them as per the literature review have been proved to be remarkable in giving the estimations of posterior probabilities. LLAIS can play an important role in solving the MSBN communication bottlenecks by facilitating inter-agent message passing and hence making it possible for realizing inference in case of full scale multi-agent probabilistic systems.

Chapter 4

Methods for Testing and Improving LLAIS

This thesis will address two concerns that are listed as follows:

1) To test the scalability and reliability of LLAIS on larger networks; since initial testing of the algorithm is done on 37 nodes network and testing beyond it has not been reported.
2) To improve the performance of LLAIS by tuning the various tunable parameters.

Hence in this chapter, we will be discussing about the methods and experimental procedure designed for testing and further improving LLAIS.

4.1 Motivation behind Testing of LLAIS

As discussed in chapter 3, LLAIS uses adaptive importance sampling to estimate posterior distribution of non-evidence nodes given the set of evidence nodes, LLAIS learns importance function sequentially to approach the optimal distribution. The prototype of LLAIS has been tested on relatively smaller network ALARM consisting of 37 nodes. The algorithm showed good convergence and accuracy on this network. LLAIS's performance did not deteriorated even when the probability of evidence was getting more and more unlikely on this network. So to check for the scalability and reliability of LLAIS it was necessary to test it on larger networks whether the algorithm performs equally well for larger networks. LLAIS is local adaptive importance sampling done on local JT in LJF; the size of local JT can vary and can be more than 37 nodes

when large BN is sectioned, so it makes it necessary to verify the performance of the algorithm.

The scalability and reliability are important factors to examine the algorithm, it means that we are checking whether the algorithm will perform equally well on the networks of variable sizes and topologies and also that if it will gives good estimates of posterior probability of non- evidence nodes as the $P(E)$ becomes more and more unlikely.

We know that in the case of unlikely evidence many of the algorithms give poor performance and convergence so it made significant for us to check if LLAIS is efficient enough to perform well in case the evidence goes more and more unlikely and also assessing the convergence of algorithm in larger networks. So far the testing of LLAIS on network with greater than 37 nodes has not been reported. Hence, we are extending the testing of LLAIS from 37 nodes network up to 109 nodes network.

We will test LLAIS on the three large networks namely – (i) Hailfinder (56 nodes) (ii) Win95pts (76 nodes) and (iii) Pathfinder (109 nodes).

4.2 Network Selection

We selected the networks larger than 37 nodes and used them for testing the performance of LLAIS. The networks are selected from Genie and Smile Bayesian repository [5]. Basically we want to test LLAIS for its scalability on the networks larger than the one used for preliminary testing [4].

For testing LLAIS, we used biggest network of size 109 nodes because LLAIS is the application of adaptive importance sampling to the LJF-local JT in MSBN. As we know that the sectioning of large Bayesian network results in the formation of subnets which denote in itself LJF-local JT in MSBN. So, the three networks selected for testing are treated as the three subnets formed as a result of sectioning of some fictitious MSBN.

It implies that our testing of LLAIS will include the subnets of sizes 56 nodes (Hailfinder network), 76 nodes (Win95pts network) and 109 nodes (Pathfinder network) denoting LJF-local JT in MSBN and will demonstrate about the scalability and affectivity of the algorithm when applied to larger local JTs in LJFs.

4.3 Experiments for Testing LLAIS

In this section we will be first discussing about the performance measures used and then the method for doing experiments for testing LLAIS for its scalability and reliability.

4.3.1 Performance Measures

Before beginning with the method for testing LLAIS, the performance measures used for computing the accuracy will be discussed.

We used Kevin Murphy's Bayesian network toolbox in MATLAB for experimenting with LLAIS. For testing of LLAIS algorithm, the exact importance function is computed and then the performance of sampling is compared with that of approximate importance function in LLAIS.

The accuracy of sampling results is preferred to be measured in terms of the distance metric named *Hellingers distance* , the reason behind it is that [49] the *Hellingers distance* weights small absolute probability differences near 0 much more heavily than similar probability differences near 1.

From [49], the *Hellinger's distance* between two distributions F_1 and F_2 which have the probabilities $P_1(x_{ij})$ and $P_2(x_{ij})$ for state j ($j = 1,2,\ldots,n_i$) of node i respectively, such that $X_i \notin \mathbf{E}$ is defined as:

$$H(F_1, F_2) = \sqrt{\frac{\sum_{X_i \in \mathbf{N}\setminus\mathbf{E}} \sum_{j=1}^{n_i} \{\sqrt{P_1(x_{ij})} - \sqrt{P_2(x_{ij})}\}^2}{\sum_{X_i \in \mathbf{N}\setminus\mathbf{E}} n_i}} \tag{4.1}$$

where **N** is the set of all nodes in the network, **E** is the set of evidence nodes and n_i is the number of states for node i. $P_1(x_{ij})$ and $P_2(x_{ij})$ are sampled and exact marginal probability of state j of node i.

In [4], for the initial testing of LLAIS, Mean Square Error (MSE) was used as the performance measure. MSE can be defined as follows:

$$MSE = \sqrt{\frac{1}{\sum_{X_i \in \mathbf{N}\setminus\mathbf{E}} n_i} \sum_{X_i \in \mathbf{N}\setminus\mathbf{E}} \sum_{j=1}^{n_i} \{P_1(x_{ij}) - P_2(x_{ij})\}^2} \tag{4.2}$$

where **N** is the set of all nodes, **E** is the set of evidence and n_i is the number of states of node i. $P_1(x_{ij})$ and $P_2(x_{ij})$ are sampled and exact marginal probability of state j of node i. But, MSE is not considered to be perfect performance measure since it has drawback that [49] it assigns equal distance for the same absolute probability difference all over the range [0,1].

As we know that probability differences near 0 are considered to be much more important than those near 1[49]. So for experiments with LLAIS we preferred using *Hellinger's distance* since it handles zero probabilities better than MSE which are very common in case of Bayesian Network [49].

4.3.2 Example for Testing Procedure

This section will explain the procedure of testing LLAIS with an example. This example will be helpful for the readers to have better understanding of the design of the experiment discussed in section 4.3.3.

Consider Hailfinder network with 56 nodes, out of the 56 nodes, 9 nodes are selected randomly as evidences denoted in Equation 4.3:

$$\boldsymbol{E} = \{e_1, e_2, e_3, \dots\dots\dots\dots, e_9\} \tag{4.3}$$

where $e_1, e_2, e_3, \dots\dots\dots\dots, e_9$ are randomly selected evidence nodes.

For simplicity of example, let the evidence nodes be instantiated using binary values where '0' will denote state of node to be false and '1' will denote true state respectively.

The evidence nodes are randomly instantiated as shown below:

$$\boldsymbol{E'} = \{e_1 = 1\ , e_2 = 0\ , e_3 = 0, \dots\dots\dots e_9 = 1\} \tag{4.4}$$

Equation 4.4 denotes $\boldsymbol{E'}$ as single **test case** representing the assignment of random values to each of the 9 evidence nodes in set.

In total, ten test cases are generated corresponding to the 9 evidence nodes set where the evidence nodes are fixed and are assigned different values randomly for ten time as can be seen below:

$$\boldsymbol{E'_1} = \{e_1 = 1\ , e_2 = 1\ , e_3 = 1\ , \ldots\ldots e_9 = 0\}$$

$$\boldsymbol{E'_2} = \{e_1 = 0\ , e_2 = 0, e_3 = 1\ , \ldots\ldots e_9 = 1\}$$

……

……

$$\boldsymbol{E'_9} = \{e_1 = 1\ , e_2 = 0\ , e_3 = 1\ , \ldots\ldots e_9 = 0\}$$

$$\boldsymbol{E'_{10}} = \{e_1 = 1\ , e_2 = 1\ , e_3 = 0\ , \ldots\ldots e_9 = 1\}$$

Now, $\boldsymbol{E'_1}, \boldsymbol{E'_2}, \ldots\ldots, \boldsymbol{E'_{10}}$ represents ten different test cases.

For each of the ten test cases, $P(E)$ is computed using exact inference method as shown below:

$$\boldsymbol{E'_1} \rightarrow P(E_1)$$

$$\boldsymbol{E'_2} \rightarrow P(E_2)$$

…. ….

… … ..

$$\boldsymbol{E'_{10}} \rightarrow P(E_{10})$$

where $P(E_1)$, $P(E_2)$ …. . $P(E_{10})$ are the exact values of probability of evidence computed using variable elimination exact inference method corresponding to each of the ten test cases generated respectively.

For each of the ten test cases,

Let $P_{exact}(x_{ij}|\boldsymbol{E}=e)$ denotes the exact value of probability for state j of node i where $x_i \notin \boldsymbol{E}$,

Let $P_{\text{approxIF}}(x_{ij}|\boldsymbol{E}=e)$ denotes the approximate value of probability for state j of node i where $x_i \notin \boldsymbol{E}$, computed from sampling using the approximate importance function in LLAIS,

Let $P_{\text{exactIF}}(x_{ij}|\boldsymbol{E}=e)$ denotes the approximate value of probability for state j of node i where $x_i \notin \boldsymbol{E}$, computed from sampling using the exact importance function in LLAIS.

For computing the accuracy of results using **approximate importance function**, *Hellinger's distance* is calculated as follows:

$$H_i(P_{\text{approxIF}}, P_{\text{exact}}) = \sqrt{\frac{\sum_{\boldsymbol{X_i}\in \mathbf{N}\setminus\mathbf{E}} \sum_{j=1}^{n_i} \{\sqrt{P_{\text{approxIF}}(x_{ij})} - \sqrt{P_{\text{exact}}(x_{ij})}\}^2}{\sum_{\boldsymbol{X_i}\in \mathbf{N}\setminus\mathbf{E}} n_i}} \quad (4.5)$$

For computing the accuracy of results using **exact importance function**, *Hellinger's distance* is calculated as follows:

$$H_i(P_{\text{exactIF}}, P_{\text{exact}}) = \sqrt{\frac{\sum_{\boldsymbol{X_i}\in \mathbf{N}\setminus\mathbf{E}} \sum_{j=1}^{n_i} \{\sqrt{P_{\text{exactIF}}(x_{ij})} - \sqrt{P_{\text{exact}}(x_{ij})}\}^2}{\sum_{\boldsymbol{X_i}\in \mathbf{N}\setminus\mathbf{E}} n_i}} \quad (4.6)$$

The estimates of posterior probabilities of non-evidence nodes from approximate and exact importance function is done after running sampling for ten times and for each run *Hellinger's distance* is computed and average of *Hellinger's distance* is calculated for each of the ten test cases.

Now we have all the required information for plotting the graph of performance. The ten test cases generated from the set of 9 evidence nodes have ten values of $P(E)$ and corresponding distance measure calculated from approximate and exact importance function respectively as shown below:

$$\boldsymbol{E'_1} = P(E_1) \rightarrow \frac{\sum_{i=1}^{10} H_i\left(P_{\text{approxIF}}, P_{\text{exact}}\right)}{10} \; and \; \frac{\sum_{i=1}^{10} H_i(P_{\text{exactIF}}, P_{\text{exact}})}{10}$$

$$\boldsymbol{E'_2} = P(E_2) \rightarrow \frac{\sum_{i=1}^{10} H_i\left(P_{\text{approxIF}}, P_{\text{exact}}\right)}{10} and \; \frac{\sum_{i=1}^{10} H_i(P_{\text{exactIF}}, P_{\text{exact}})}{10}$$

……..

……

$$\boldsymbol{E'_9} = P(E_9) \rightarrow \frac{\sum_{i=1}^{10} H_i\left(P_{\text{approxIF}}, P_{\text{exact}}\right)}{10} \; and \; \frac{\sum_{i=1}^{10} H_i(P_{\text{exactIF}}, P_{\text{exact}})}{10}$$

$$\boldsymbol{E'_{10}} = P(E_{10}) \rightarrow \frac{\sum_{i=1}^{10} H_i\left(P_{\text{approxIF}}, P_{\text{exact}}\right)}{10} \; and \; \frac{\sum_{i=1}^{10} H_i(P_{\text{exactIF}}, P_{\text{exact}})}{10}$$

The same steps are followed for computing MSE and plotting the graph for analyzing the performance of LLAIS.

To summarize the procedure, we randomly selected a set of 9 evidence nodes, assigned these fixed set of evidences different values randomly for ten times resulting in ten test cases. Then for each test case, compute the exact value of probability of evidence and compute the accuracy of results produced from approximate importance function and exact importance function in terms of *Hellinger's distance* by running sampling for ten times and averaging the *Hellinger's distance* corresponding to each test case. Lastly, plot the graph to observe the performance of LLAIS as the value of evidence goes more and more unlikely.

The procedure discussed in this section is by taking a set of 9 evidence nodes which denotes the first sequence of 10 test cases.

Likewise, second and third sequences are generated consisting of ten test cases each respectively.

The three sequences generated for each network being tested are described as follows:

- **First Sequence** = 9 evidence nodes are fixed in a set and are given random values for ten times indicating the generation of ten test cases corresponding to the respective sequence of 9 evidence nodes.
- **Second sequence** = 11 evidence nodes are fixed in a set and are given random values for ten times indicating the generation of ten test cases corresponding to the respective sequence of 11 evidence nodes.
- **Third sequence** = 13 evidence nodes are fixed in a set and are given random values for ten times indicating the generation of ten test cases corresponding to the respective sequence of 13 evidence nodes.

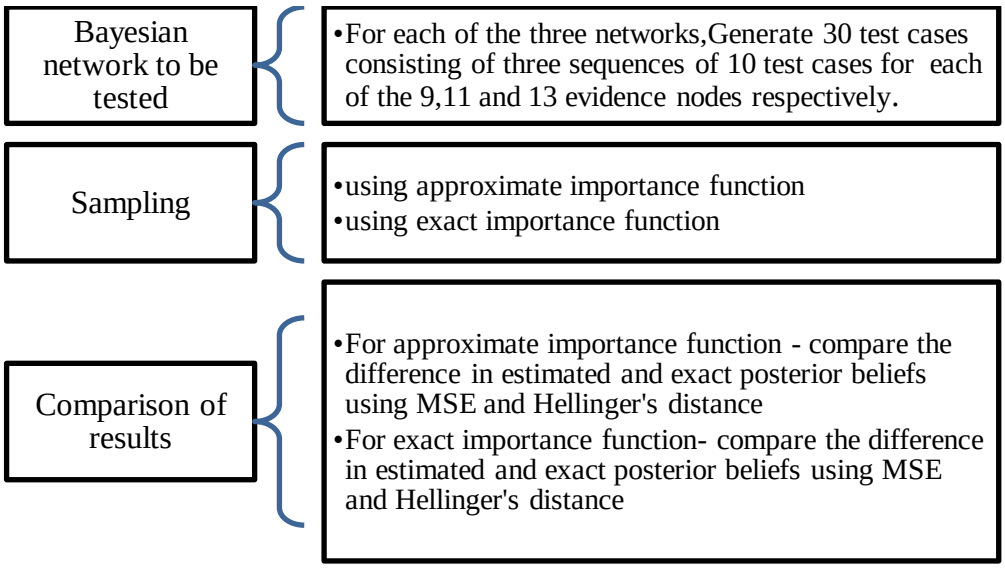

Figure 4.1: Procedure for testing LLAIS.

Total Test cases generated for each of the three networks tested = 30

(3 sequences × 10 test cases (from each sequence respectively))

Figure 4.1 discuss in brief the general procedure for testing LLAIS on the three networks. The test cases are plotted on graph for the analyzing the performance of LLAIS.

4.3.3 Design of Experiment

Now, in this section the design of the experiment for testing LLAIS will be discussed.

Step 1*: Downloading three networks named Hailfinder (56 nodes), Win95pts (76 nodes) and Pathfinder (109 nodes) from Genie and Smile and converting it to the format acceptable in MATLAB.*

Step 2*: For each of the three networks, randomly generating a set of evidence nodes. The evidence nodes are fixed in the set and are given random values defining a test case. The assignment of random values to the set of fixed evidence nodes is done for ten times respectively to generate ten test cases.*

Step 3*: Calculating the exact value of $P(E)$ using exact inference method corresponding to each of the ten test cases.*

Step 4: *Computing $P_{approxIF}(x_{ij}|E=e)$ using approximate importance function in LLAIS and $P_{exactIF}(x_{ij}|E=e)$ using exact importance function, for state j of node i where $x_i \notin \boldsymbol{E}$ respectively for each of the ten test cases.*

Step 5: *Computing the Hellinger's distance and Mean Square Error (MSE) for the values of $P_{approxIF}(x_{ij}|E=e)$ and $P_{exactIF}(x_{ij}|E=e)$ for each of the ten test cases generated respectively to compare the estimates of posterior beliefs from approximate importance function and exact importance function.*

***Step 6**: Plotting the graph showing relation between the performance measures computed in step 5 and $P(E)$ estimated in step 3.*

The above 6 steps are followed for three sequences of 9, 11 and 13 evidence nodes set respectively generating in total of 30 test cases and hence analyzing the performance of LLAIS.

4.4 Improving LLAIS by Tuning the Tunable Parameters

So far the testing of LLAIS as discussed in the Section 4.1 has been done on 37 nodes network and it showed good results for estimating the posteriors. But apart from testing it is equally important to improve the algorithm. The second part of the thesis deals with the tuning of parameters to improve the performance of LLAIS. There are many tunable parameters in LLAIS such as the heuristic value of **threshold** $\in -\boldsymbol{cutoff}$, updating intervals, number of updates, number of samples, learning rate discussed as follows:

1. Threshold $\in -\boldsymbol{cutoff}$ is used for handling very small probabilities in the network. For example, from [45], if the root node with state s has prior probability $Pr(s) = 0.0001$ and the posterior probability is given by $\Pr(s|E) = 0.8$, from the simple calculation we can assume that if we update the importance function with 1000 updating interval then we can think of hitting s only one time after ten updates so it can result in slow convergence. So this problem is overcome by setting the threshold value θ and replacing every probability $p < \theta$ by θ. As discussed in [45] the convergence rate is quite sensitive to this threshold and it is important to set for its proper value.

Since the distribution of probabilities is different for all the networks and also the most extreme probabilities in these networks make the inference task more difficult, so the role of threshold cut-off becomes important. It is equally very important to set and adjust these cut-off values to get better performance output. This technique of $\in -cutoff$ heuristic as discussed earlier was originally proposed in [45] and it asserted that the smaller threshold might lead to slow convergence in some cases and faster in others, so if one threshold does not work well we can change it to the specific value for improving the convergence.

The proper tuning helps the tail of importance function not to decay faster, the optimal value for $\in -cutoff$ is dependent upon the network, so in [49] experiments are done by giving different cutoff values to the nodes with different number of outcomes. Hence the heuristic cutoff value depends upon the nature of distribution of network for better performance. Similarly it also plays vital role to get less error and better precision.

We performed experiments on three networks, that is, Hailfinder, Win95pts and Pathfinder network, relatively the most extreme probabilities were encountered in Pathfinder and Hailfinder and so the threshold value had to be adjusted to improve the performance of LLAIS. From the empirical testing on the three networks we could not determine any universal value of threshold which can always yield better results for every network which is to tested, since if one heuristic cutoff works well on one network it might not work on the other. So the nature of distribution of probabilities in the network plays role in selection of $\in -cutoff$. From the various experimentations done on the

three networks, we recommend to use $\epsilon = 0.01$ for the nodes with the number of outcomes less than 5, $\epsilon = 0.006$ for nodes with the number of outcomes between 5 and 8, otherwise $\epsilon = 0.0005$. These experiments with different cutoff values were motivated from [49].

2. The next tunable parameter is the **number of updates** and **updating interval**. The number of updates plays an important role in the sense that it denotes how many times scores in table have to be updated so that they give us optimal output and updating interval denotes the number of samples that have to be updated. The proper value of updating interval and number of updates is crucial, if the value of updating interval is small it may result in time consuming process as the small set of samples may have to be updated many times to get the optimal value. LLAIS uses five updates and taking small set of 2000 samples which is quite time consuming to update 5 times and we believe that better tuning of parameters can result in getting better output in less updates. From our tuning of parameters and many empirical tests we fixed the value of updating interval to be 2100 and it requires only three updates to give better results as compared to LLAIS with five updates.

3. The **number of samples** is a very important parameter, since in the stochastic sampling algorithm the performance of the algorithm increases as the number of samples increases but it is always good to have minimum number of samples that can help you reach better output and hence making the sampling process time efficient. In original LLAIS, [1] they have used 5000 samples but with the proper tuning of other parameters

we are able to reduce this number to 4500 and getting much better performance in comparison to the original LLAIS.

4. **The learning rate** in [45] is defined as the rate at which optimal importance function will be learned as per the formula $\eta(k) = a(\frac{a}{b})^{k/kmax}$, where a = initial learning rate, b = learning rate in the last step, k = number of updates, and $kmax$ = total number of updates. For LLAIS, the values were set as a=0.4 and b=0.14. The value of learning rate should also be tuned well because if the value of learning rate is small then the algorithm takes too much time to converge and if its large then the algorithm starts showing divergence. We did not change these values and kept the learning rate same as [4] with a =0.4 and b=0.14.

The tunable parameters are tuned after many experiments in which they were given heuristically different values and then checked for their performance. Since the values of thresholds have to be fixed in according to the probability distribution of the CPTs so it means every network requires different values of tunable parameters. After the experimentation with different values of tunable parameters, the values of tunable parameter are finalized in such a way that they give good result in terms of improved accuracy and take less time for all the three networks tested.

Table 4.1 below shows comparison of the values of tunable parameters for original LLAIS algorithm and the improved LLAIS.

Tunable parameters	Original LLAIS	Improved LLAIS
Number of samples	5000	4500
Number of updates	5	3
Updating interval	2000	2100
Threshold value	Nodes with outcomes <5 = 0.05 Nodes with outcomes < 8 = 0.005 Else = 0.0005	Nodes with outcomes < 5 = 0.01 Nodes with outcomes < 8 = 0.006 Else = 0.0005

Table 4.1: Shows the comparison of values of various tunable parameters for original LLAIS and improved LLAIS.

4.5 Methods for Testing the Improved LLAIS

In this section the design of experiments for comparing the original LLAIS and improved LLAIS will be discussed along with the performance measure used.

4.5.1 Design of Experiment

For each of the three networks, that is, Hailfinder, Win95pts and Pathfinder - we generated three sequences each containing ten test cases of 9, 11 and 13 evidence nodes set respectively.

The steps for experiments are same as followed during the testing of LLAIS starting from downloading of networks then generating ten test cases corresponding to the set of 9, 11

and 13 evidence nodes. Then computing the exact value of P(E) for each of the ten test cases respectively. Then, computing $P_{org}(x_{ij}|\boldsymbol{E}=e)$ *from original LLAIS without tuning of parameters and* $P_{imp}(x_{ij}|\boldsymbol{E}=e)$ *from LLAIS improved after tuning the parameters, for state* j *of node* i *where* $x_i \notin \boldsymbol{E}$ *respectively for each of the ten test cases. Then calculating the Hellinger's distance using* $P_{org}(x_{ij}|\boldsymbol{E}=e)$ *and* $P_{imp}(x_{ij}|\boldsymbol{E}=e)$ *respectively for each of the ten test cases generated respectively to compare the estimates of posterior beliefs from original LLAIS and LLAIS improved and finally plotting the graph to analyze the performance of original LLAIS and improved LLAIS.*

We expect that after tuning the parameters LLAIS will become time efficient for now it requires less number of samples and less updates for learning the optimal importance function, that is, now improved LLAIS in 4500 samples instead of 5000 gives better results with number of updates = 3 instead of 5 to learn the optimal distribution and also it is expected that it will give more accurate results since the threshold values for dealing with small probabilities have been tackled more properly.

4.5.2 Performance Measure

For comparing the performance of original LLAIS and improved LLAIS, *Hellinger's distance* is preferred. As discussed in section 4.3.3, *Hellinger's distance,* handles zero probabilities in BN more efficiently than MSE.

4.6 Discussion

In this chapter we discussed about the methods and design of experiments for testing and

improving LLAIS. The experiment procedure to test LLAIS for its scalability and reliability are discussed in detail starting from the making of networks in MATLAB to applying performance measures for computing accuracy. Since we know that LLAIS has many tunable parameters so these parameters are tuned and now there is a possibility that tuning of parameters will result in improving the algorithm and producing better accuracy.

Chapter 5

Experiment Results for Testing and Improving LLAIS

In this chapter, the experiment results for testing LLAIS on the three networks will be discussed. Also there are various tunable parameters in LLAIS as discussed in chapter 4, these parameters are tuned and experiment results in this chapter will demonstrate that LLAIS properly tuned shows significant improvement in the performance.

5.1 Testing of LLAIS

In this section, LLAIS will be tested on large networks to check for its scalability and reliability. In the case of unlikely evidence many of the approximate algorithms give poor performance so it is significant to check if LLAIS is efficient enough to perform well in case of unlikely evidence and also to assess the convergence of the algorithm on large networks.

As discussed in chapter 4, the testing of LLAIS is done on three networks-
(i) Hailfinder (56 nodes) (ii) Win95Pts (76 nodes) (iii) Pathfinder networks (109 nodes).

For the testing of LLAIS algorithm, the exact importance function is computed so that it is easy to determine how close is the performance of approximate importance function in LLAIS to the exact importance function.

In order to compare the accuracy of sampling using the exact and approximate importance we calculated their departure from the exact solution obtained using variable elimination method.

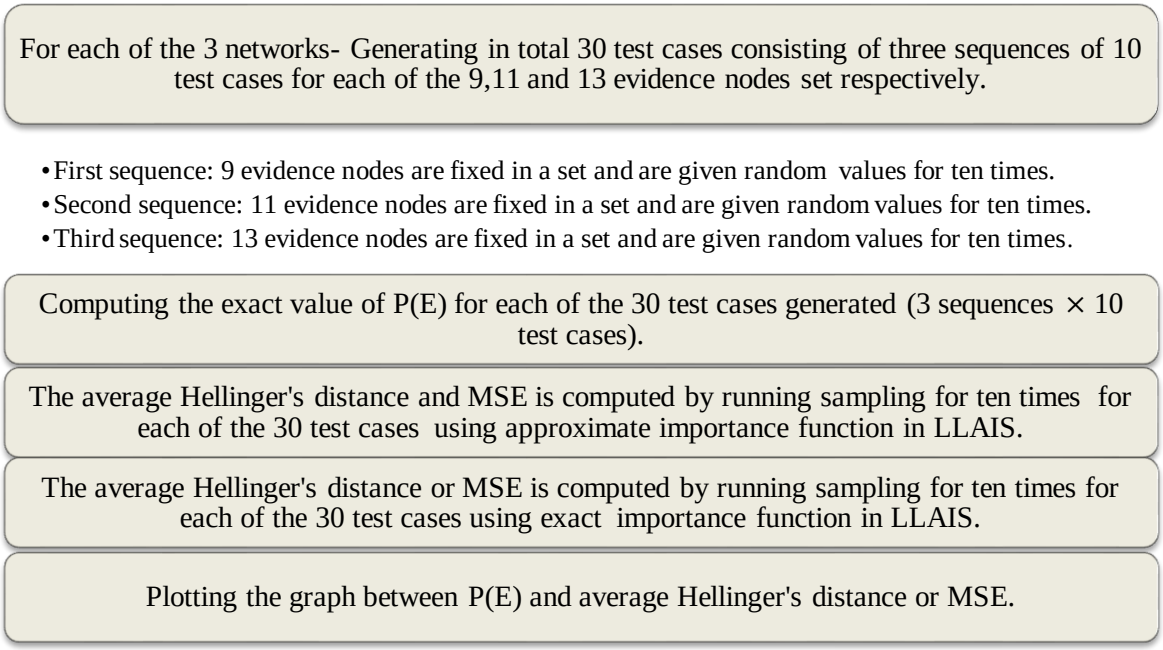

Figure 5.1: Steps for testing LLAIS on larger networks.

The Figure 5.1 explains diagrammatically the general procedure of experiments for testing LLAIS on larger networks in brief.

5.2 Experiment Results

The following section will discuss in detail about the experiment results for testing LLAIS on each of the three networks to determine how close the approximate importance function (LLAIS) is able to reach optimal results.

5.2.1 Testing Results on Hailfinder BN (56 nodes network)

The first network tested for LLAIS performance is The Hailfinder Network.

The Hailfinder network contains 56 nodes representing 56 discrete random variables, this network was designed for forecasting severe summer hail in northeastern Colorado[54].

The following Figure 5.2 shows the topology of the network.

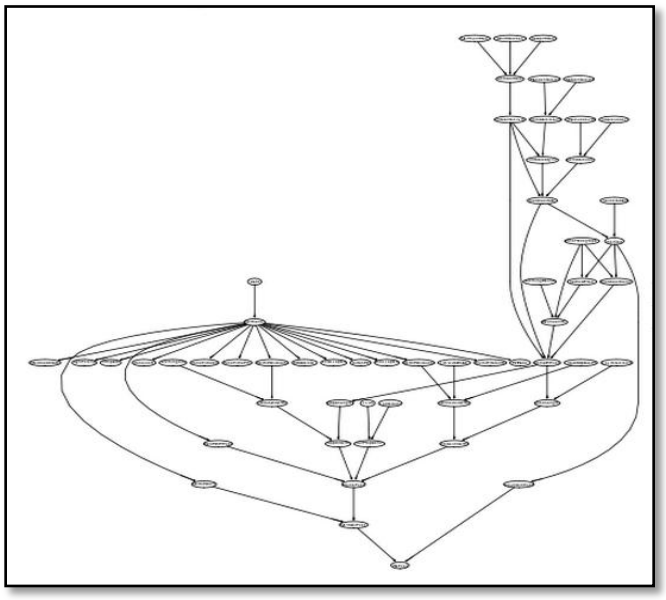

Figure 5.2: [5] Structure of Hailfinder Bayesian network with 56 nodes and 66 edges.

The following Figure 5.3 represents the graph for the performance of sampling using approximate importance function in LLAIS and the exact importance function in terms of *Hellinger's distance* with 9 evidence nodes set. As discussed in chapter 4, procedure involves assigning random values to the fixed set of 9 evidence nodes for ten times and computing exact value of $P(E)$ for each time. Then calculating the average *Hellinger's distance* corresponding to each test case respectively after running LLAIS for ten times in

each test case. It should be noted that $P(E)$ plotted in the graphs for all the networks is calculated by using exact inference method to compare the accuracy of the estimates obtained using LLAIS.

The graph in Figure 5.3 is drawn representing the relation between *Hellinger's distance* and $P(E)$. The *Hellinger's distance* is plotted on y-axis while $P(E)$ is plotted on the x-axis. The $P(E)$'s are arranged in the descending order ranging from the most likely evidence to the most unlikely evidence. Then the average *Hellinger's distance* is plotted against each $P(E)$ respectively. As observed from the graph the most likely evidence from the set of 9 evidence nodes is 1.97e-05 while the most unlikely is 1.88e-07. It can be seen in the graph that the performance of approximate importance function in LLAIS is really good when compared to the exact importance function output.

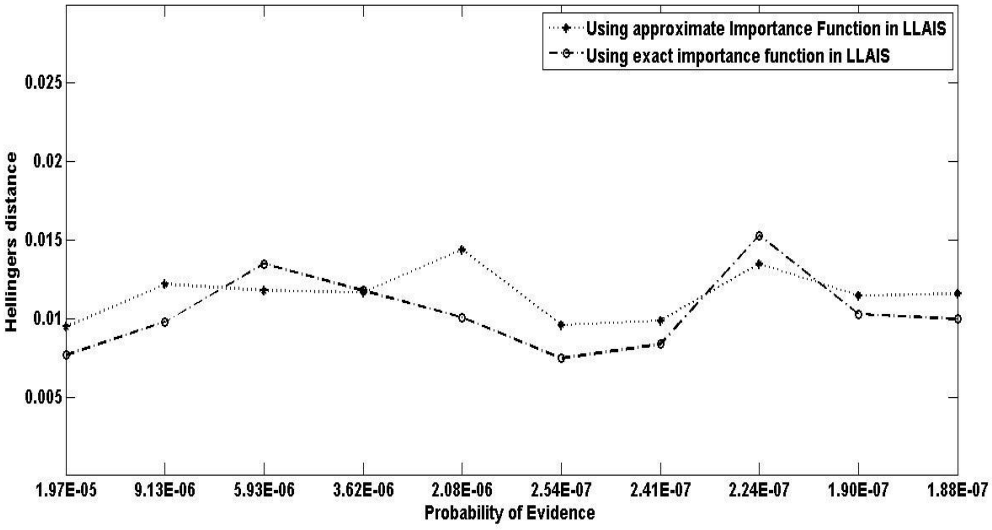

Figure 5.3: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Hailfinder network with 9 evidence nodes.

Similarly, following the same procedure for doing experiments as discussed in section 5.1, the second sequence of ten test cases consisting of 11 evidence nodes set is generated. The graph is plotted in the same way as done for the set of 9 evidence nodes in previous case between the $P(E)$ and *Hellinger's distance* as shown in Figure 5.4 by taking $P(E)$ along x-axis marking its range from the most likely to unlikely evidence and *Hellinger's distance* along y-axis.

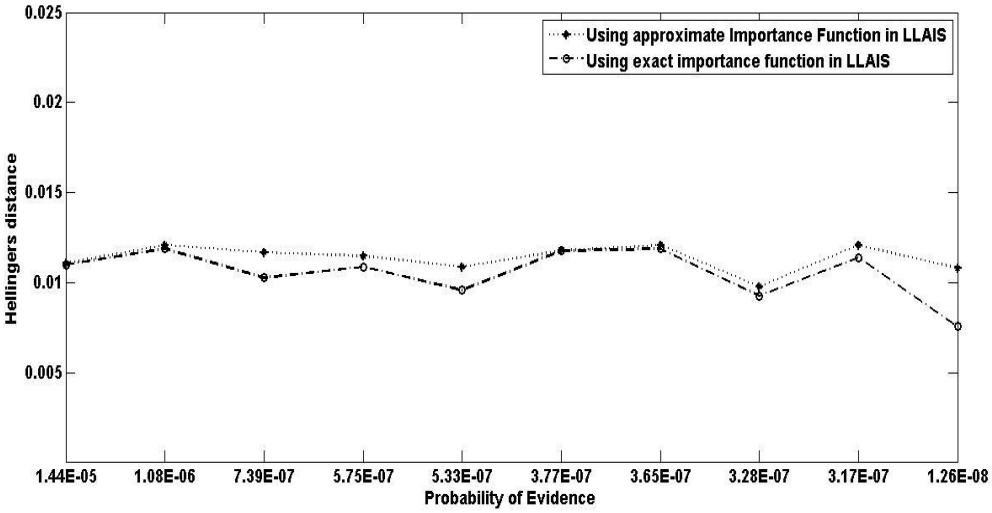

Figure 5.4: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Hailfinder network with 11 evidence nodes.

From the graph in Figure 5.4, it can be seen that the most likely evidence encountered for this evidence set is 1.44e-05 while the most unlikely evidence is 1.26e-08. As seen in the graph, sampling accuracy using the approximate importance function in LLAIS is quite comparable to the results obtained using the exact importance function.

Figure 5.5 represents the graph drawn for the analyses of performance of LLAIS when third sequence of 13 evidence nodes set is taken. The graph is plotted from the output

showing relation between *Hellinger's distance* and $P(E)$ in order to determine the performance of LLAIS as the evidence goes unlikely, it can be seen in the graph the most likely evidence for this case is 2.31e-07 while unlikely evidence is 3.57e-11. The performance of LLAIS using approximate importance function is achieving almost the same accuracy as achieved by exact importance function.

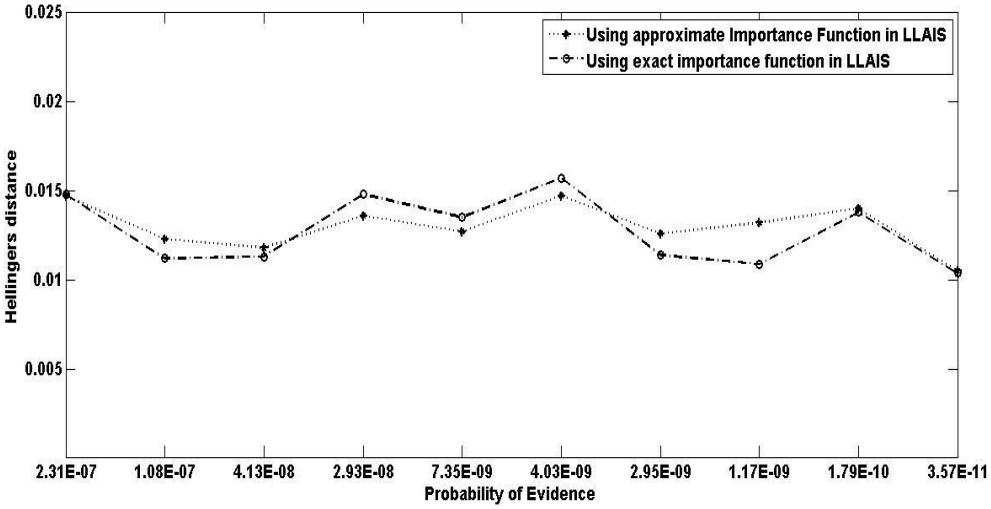

Figure 5.5: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Hailfinder network with 13 evidence nodes.

From the analysis of graphs drawn using 9, 11 and 13 evidence nodes set for Hailfinder network we can say that the performance of sampling by using approximate importance function in LLAIS is somewhat same as that of using exact importance function. The only drawback we can talk here about using approximate importance function is that it is time inefficient because it requires updating and learning of optimal distribution.

Figure 5.6 represents the histogram which displays frequency distribution of $P(E)$ for the all the 30 test cases generated for Hailfinder network, where we took three sequences of

10 test cases each. The histogram shows the information about the order of unlikeliness of evidence generated from our randomly selected evidence set of 9,11 and 13 evidence nodes. From the histogram it is easily observed that the most frequently occurring $P(E)$ from the evidence set we generated is of the order of 10^{-7} for Hailfinder network and most unlikely of the order of 10^{-10} and 10^{-11}.

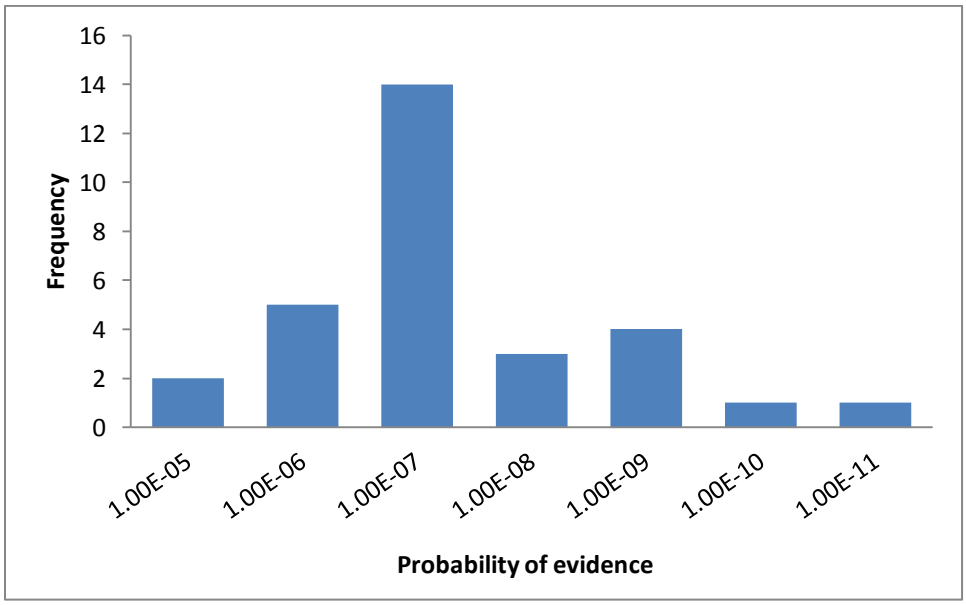

Figure 5.6: Frequency distribution of P(E) in Hailfinder network.

The following Table 5.1 shows the statistical information of the results by combining all the 30 test cases generated during the experimentation for testing LLAIS. As seen in the table the maximal *Hellinger's distance* from approximate importance function is 0.0147 and from the exact importance function is 0.0157 which is more than the former. Also the minimal *Hellinger's distance* using approximate importance function in LLAIS is 0.0095 while for the exact value of importance function is 0.0075. The variance using exact importance function is more in comparison to approximate importance function; hence

we can say that LLAIS is quite scalable for this network since its sampling results vary less in comparison to the results obtained using exact importance function.

Hellinger's distance	Using approximate Importance Function	Using Exact Importance Function
Minimum Error	0.0095	0.0075
Maximum Error	0.0147	0.0157
Median	0.0118	0.0111
Variance	1.99e-06	4.92e-06
Mean	0.0118	0.0113

Table 5.1: Statistical results for all the 30 test cases generated to test LLAIS for Hailfinder network in terms of *Hellinger's Distance*.

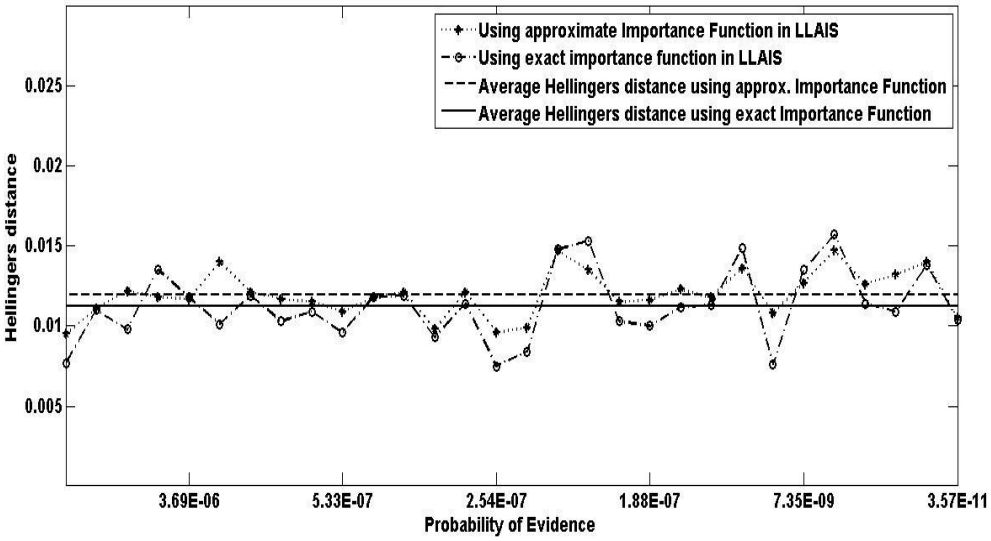

Figure 5.7: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Hellinger's distance* for Hailfinder network.

Since we wanted to make the comparison between the performances measures *Hellinger's distance* and Mean Square Error (MSE) so MSE is also calculated for all the test cases with same evidence nodes sets and same values.

The maximal MSE computed using exact importance function is 0.0146 while using approximate importance function is 0.02 which is much more than the former. Also the minimal MSE obtained using approximate importance function is 0.0082 and from the exact importance function in 0.0071. The variance for exact importance function is little less than using approximate function, the value of variance from the results of approximate importance function is 4.64e-06 using MSE while from *Hellinger's distance* as seen earlier in Table 5.2 was 1.99e-06, which shows difference in capturing of errors by two performance measures. As discussed in chapter 4 we can interprete that accuracy obtained from *Hellinger's distance* is much more accurate than from MSE.

Mean Square Error	Approximate Importance Function	Exact Importance Function
Minimum Error	0.0082	0.0071
Maximum Error	0.02	0.0146
Median	0.0103	0.0104
Variance	4.65e-06	4.10e-06
Mean	0.0107	0.0106

Table 5.2: Statistical results for all the 30 test cases generated to test LLAIS for Hailfinder network in terms of *Mean Square Error*.

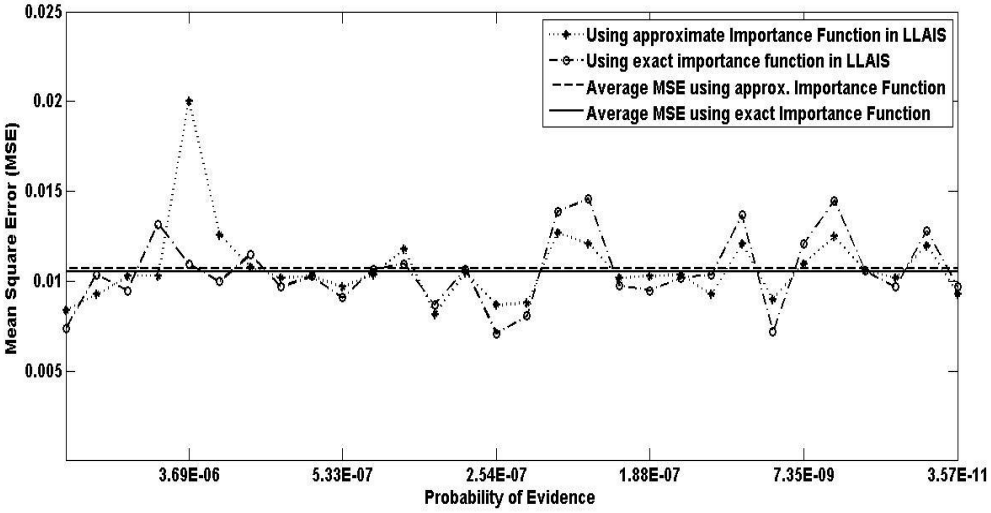

Figure 5.8: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Mean Square Error* for Hailfinder network.

Hence from the experimentation of testing on Hailfinder network, it can be concluded that the approximate importance function in LLAIS performs quite good when compared with the performance of sampling from the exact importance function. We can say that the LLAIS is scalable and reliable to be applied on this network even when $P(E)$ goes unlikely.

5.2.2 Testing Results on Win95pts BN (76 nodes network)

The Win95pts Bayesian network containing 76 random variables is developed by Microsoft Research is basically an expert system which was developed for trouble shooting in the Windows 95[5].

The experiment for testing is performed in the similar way as discussed in the last section 5.2. In total 30 test cases are generated with three sequences of ten test cases each taking

9, 11 and 13 evidence nodes for each of the three sequences respectively. The graph is plotted between $P(E)$ and *Hellinger's distance* for each of the ten test cases respectively.

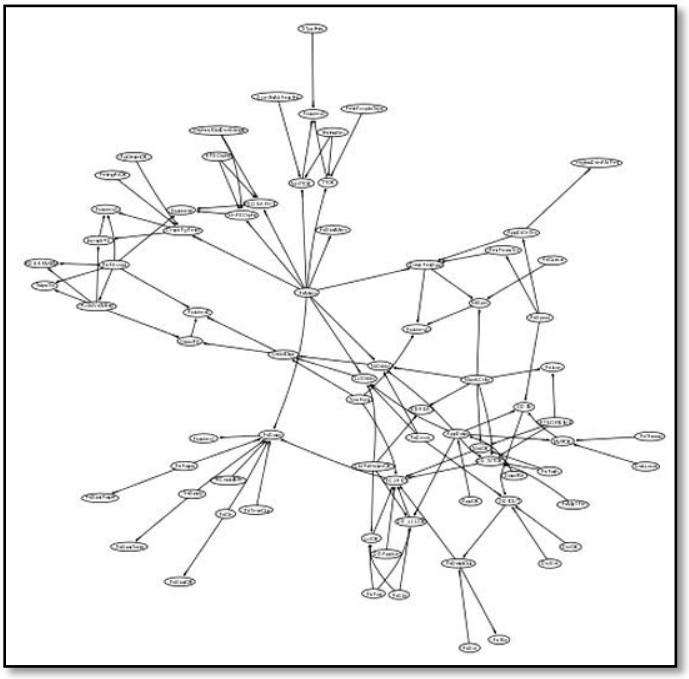

Figure 5.9 : [5] Structure of win95pts Bayesian network
with 76 nodes and 112 edges.

The first sequence of ten test cases is generated where the set of evidence nodes consists of 9 evidence nodes and for ten times these fixed 9 evidence nodes are given random values. The graph is plotted where $P(E)$ is plotted against x-axis with the *Hellinger's distance* plotted along y-axis in the same way as previous graphs drawn is case of Hailfinder network. For the set of 9 evidence nodes, as can be seen in Figure 5.10, the most likely evidence encountered is 6.12e-03 while the most unlikely evidence is 8.99e-17. From the graph it can be determined that performance of approximate importance

function in LLAIS is still not bad when compared with the sampling results from exact importance function and is showing results quite close to the exact importance function.

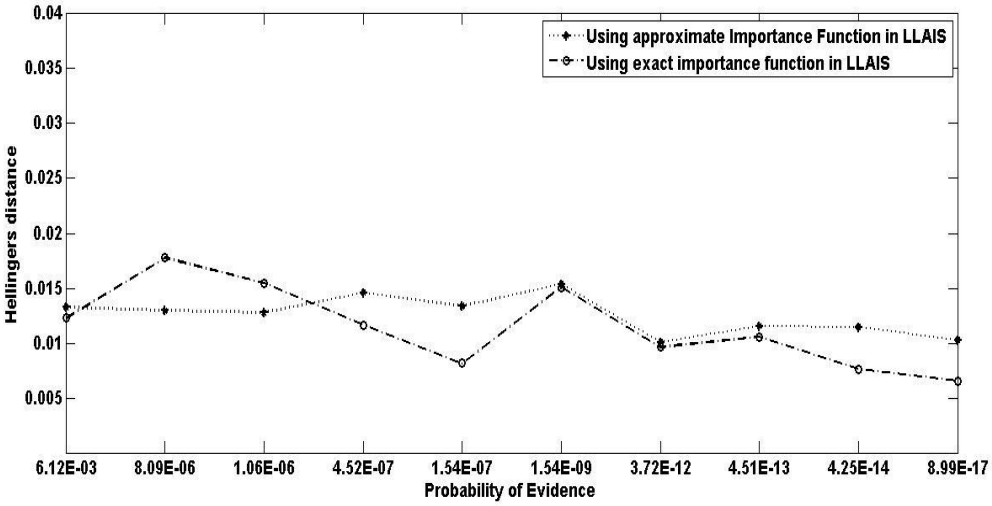

Figure 5.10: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Win95pts network with 9 evidence nodes.

Similarly, for the second sequence consisting of ten test cases from 11 evidence nodes, the graph is plotted between the *Hellinger's distance* and $P(E)$ as shown in Figure 5.11.

As seen in the graph, the most likely evidence obtained from the set of 11 evidence nodes is 4.61e-02 while the most unlikely evidence is 2.67e-14. For this case as we can observe in the graph that the sampling done using the exact importance function shows better results than using approximate importance function in LLAIS but still we can say that the results from the approximate importance function are still not that bad as compared to the results obtained using the exact importance function.

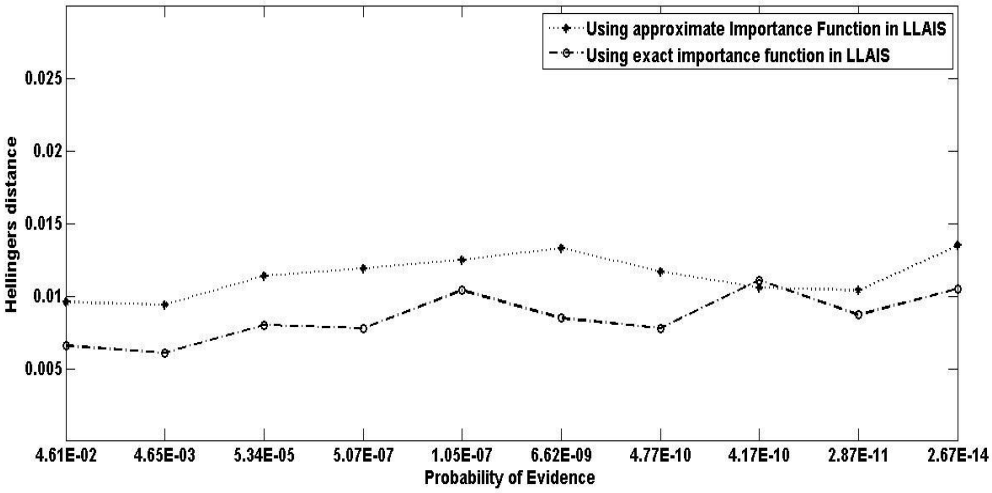

Figure 5.11: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Win95pts network with 11 evidence nodes.

Now the third sequence of ten test cases consisting of 13 evidence nodes set is generated. The graph is plotted in the same way as done in the earlier cases shown in the Figure 5.12, the exact value of $P(E)$ is calculated for 10 times and average of *Hellinger's distance* is plotted corresponding to each $P(E)$ after running LLAIS using approximate and exact importance function for ten times. By analyzing the graph in Figure 5.12 we can see that the most likely evidence observed is 2.73e-07 and most unlikely evidence is 5.34e-13. It is also viewed in graph that exact importance function in LLAIS gives better performance as compared to the approximate importance function in LLAIS.

Hence, from the graphs drawn for the set of 9, 11 and 13 evidence nodes it can be observed that the approximate importance function in LLAIS almost reaches the same accuracy given by exact importance function. So in case of Win95pts network the

performance of approximate importance function can be taken comparable to the exact importance function and hence can be regarded as scalable for this network too.

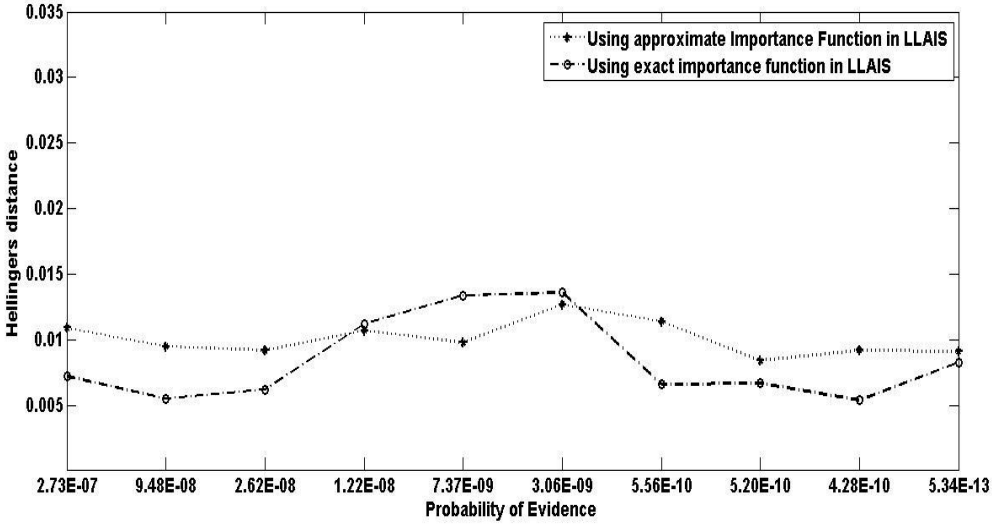

Figure 5.12: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Win95pts Network with13 evidence nodes.

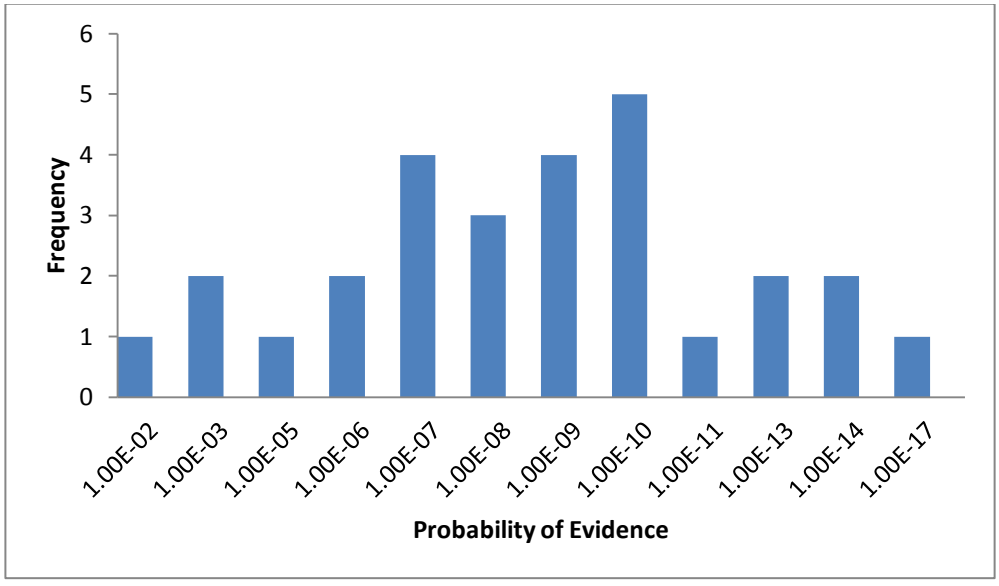

Figure 5.13: Frequency distribution of P(E) in Win95pts network.

In order to observe the frequency distribution of the order of unlikeliness of $P(E)$ generated, the histogram is plotted as it can be seen in the Figure 5.13. From the analysis of the histogram shown below it can be interpreted that from the randomly selected evidence nodes the most frequently occurring $P(E)$ for this network is of the order of 10^{-10}.

In order to make comparison between the performance measures MSE is also computed in addition to *Hellinger's distance*.

The following Tables 5.3 and 5.4 shows the difference in capturing of error by the two performance measures. The Table 5.3 below shows the statistical information of the performance measured using *Hellinger's distance* from the combination of all the 30 test cases.

Hellinger's distance	Using Approximate Importance Function	Using Exact Importance Function
Minimum Error	0.0084	0.0054
Maximum Error	0.0154	0.0178
Median	0.0114	0.0084
Mean	0.0114	0.0095
Variance	3.18e-06	1.03e-05

Table 5.3: Statistical results for all the 30 test cases generated for Win95pts network in terms of *Hellinger's distance.*

As seen in the Table 5.3 the maximal *Hellinger's distance* from the approximate importance function is 0.0154 while from the exact importance function is 0.0178 almost comparable to each other. The minimal *Hellinger's distance* from the exact importance function is 0.0054 and from the approximate importance function 0.0084 hence not much difference between the two. On the other hand, the variance for exact importance function is much more than that of using approximate importance function which shows that approximate importance function in LLAIS gives better performance for this network in terms of *Hellinger's distance.*

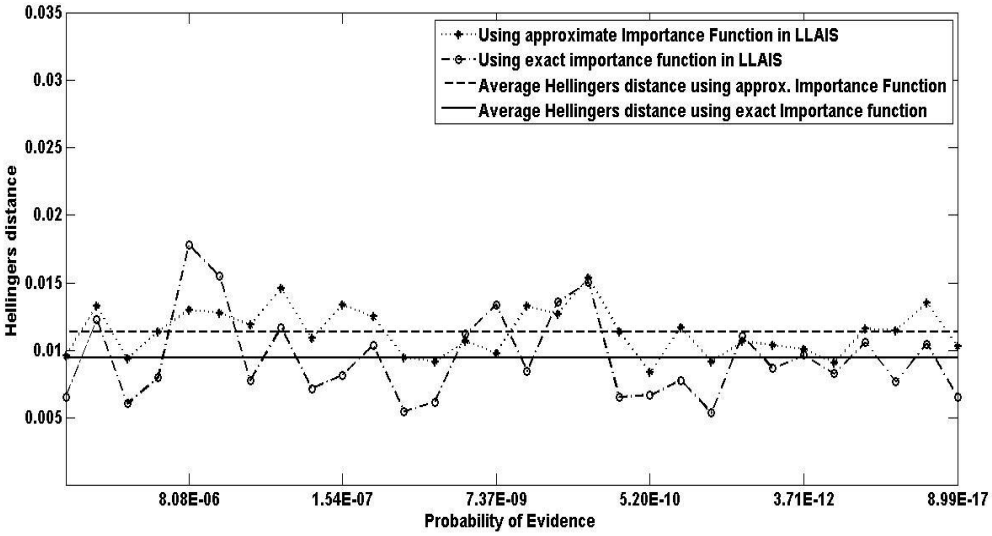

Figure 5.14: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Hellinger's distance* for Win95pts network.

Similarly, calculating the statistical measures for the sampling results in terms of MSE. The Table 5.4 below shows the performance of approximate and exact importance function in LLAIS in terms of MSE.

Mean Square Error	Approximate Importance Function	Exact Importance Function
Minimum Error	0.0086	0.0046
Maximum Error	0.0162	0.0154
Median	0.0113	0.0077
Mean	0.0113	0.0086
Variance	2.56e-06	8.90e-06

Table 5.4: Statistical results for all the 30 test cases generated for Win95pts network in terms of *Mean Square Error.*

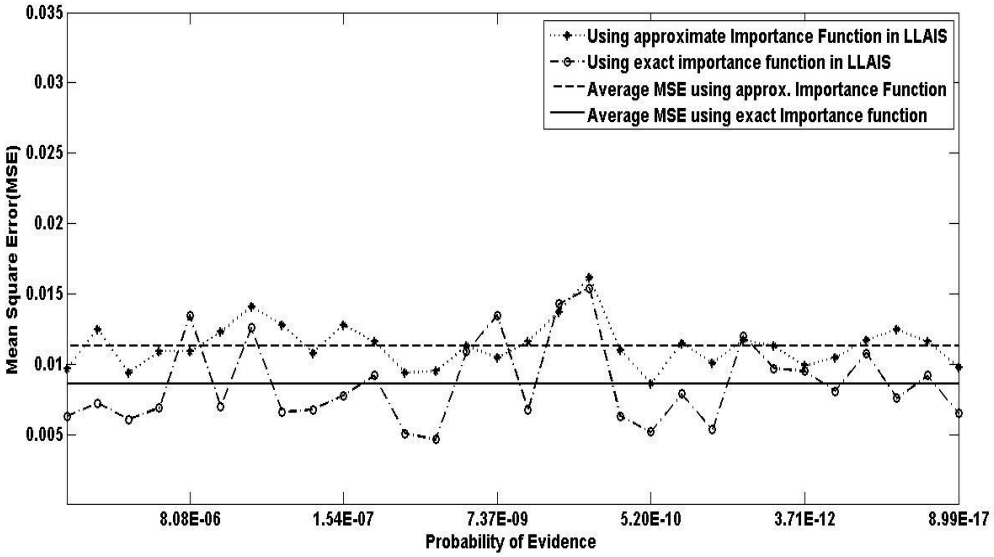

Figure 5.15: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Mean Square Error* for Win95pts network.

As seen in Table 5.4 the maximal MSE for approximate importance function in LLAIS is 0.0162 while for the exact importance function it is 0.0154. The minimal MSE using approximate importance function is 0.0086 and from the exact importance function is 0.0046. The maximal and minimal values of MSE are almost comparable for both the

importance functions. The variance recorded by MSE is more for sampling results using the exact importance function.

To summarize the testing experiments on Win95pts network, we can say that the approximate importance function in LLAIS performs quite good and comparable to the exact importance function. As seen in the graphs in Figure 5.14 and Figure 5.15 approximate importance function in LLAIS shows less in the variance in results in comparison to the exact importance function. So LLAIS using approximate importance function can be regarded as scalable and reliable for this network too.

5.2.3 Testing Results on Pathfinder BN (109 nodes network)

The third network tested is The Pathfinder network consisting of 109 nodes. It is basically an expert system which is created to assist surgical pathologists in the diagnosis of lymph-node disease [55]. The Figure 5.16 shows the structure of Pathfinder taken from Norsys Software Corp [56].

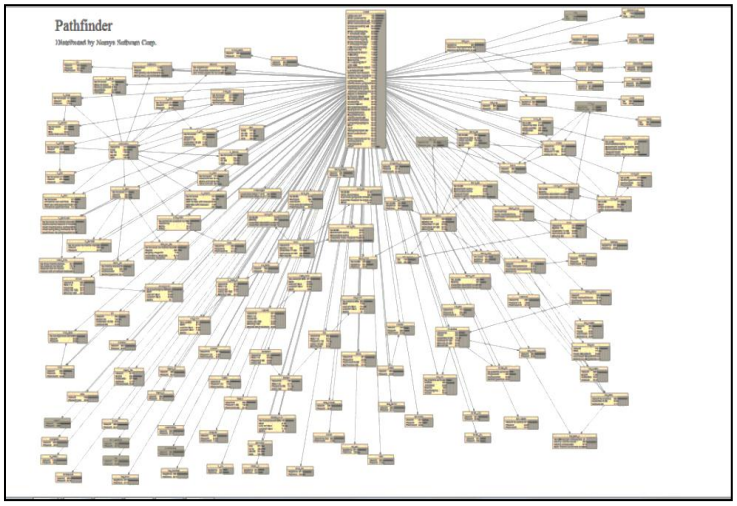

Figure 5.16: Structure of Pathfinder Bayesian network from Norsys Software Corp.

The experiments for testing LLAIS on Pathfinder network are performed in the same way as in case of Hailfinder and Win95pts network. The sampling output from approximate importance function in LLAIS is compared with that of exact importance function. Following the same procedure in total 30 test cases are generated for this network too consisting of three sequences of ten test cases each of the 9, 11 and 13 evidence nodes respectively. Then *Hellingers' distance* is calculated to measure the performance of LLAIS.

Figure 5.17 shows the comparison of efficacy of sampling using approximate and exact importance function for 9 evidence nodes set in Pathfinder network. The most likely evidence for this evidence set is 3.76e-03 while the most unlikely is 4.22e-39 which seems to be quite extreme as compared to the unlikely evidences encountered in other two networks. So this network can be thought as challenging enough for testing LLAIS because of the presence of extreme probabilities.

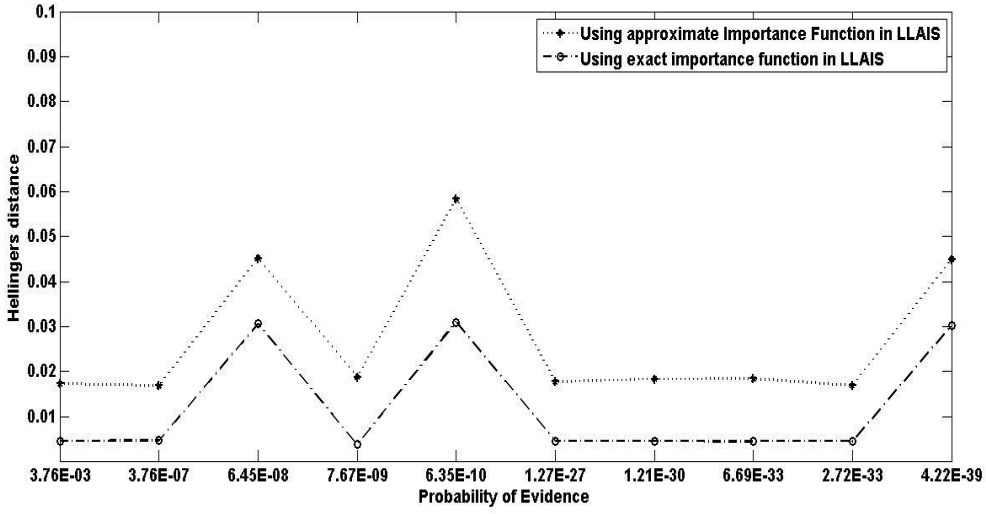

Figure 5.17: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Pathfinder network with 9 evidence nodes.

It can be seen in Figure 5.17 that the performance given by LLAIS is different from the other two networks showing lots of ups and down in the accuracy. The reason behind this can be generation of bad samples for some cases and for some dealing with extreme probabilities. Since it can be seen in the graph the range of $P(E)$ shows lot of variation in comparison to the other two networks so graph is showing strange trend in the sampling results.

In this case the approximate importance function in LLAIS did not give good results as it gave in other two networks tested earlier. The performance of exact importance function is also not good since there is lot of variance obtained but still the exact importance function performs better in this case.

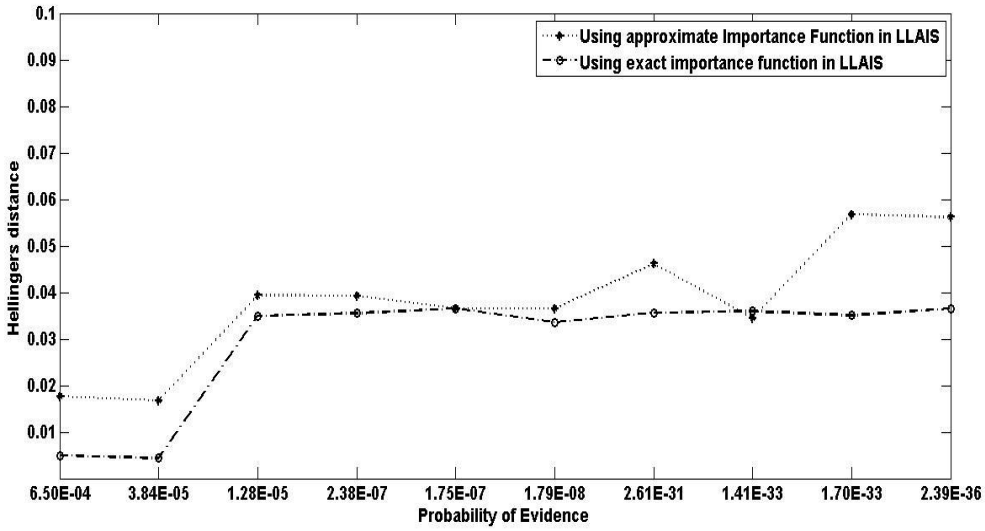

Figure 5.18: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Pathfinder network with 11 evidence nodes.

Figure 5.18 shows the graph for the set of 11 evidence nodes and it can be seen that the most likely evidence encountered for this case is 6.50e-04 and the most unlikely evidence

is 2.39e-36. Hence in this case also it is seen the range of order of unlikeliness of evidence is too extreme hence here also the performance of sampling will show lots of variance. From the graph in Figure 5.18, it can be observed that the performance of approximate importance function is again not too good in comparison to the exact importance function. The exact importance function somewhat converges even when the evidence goes more unlikely but it did not happened in case of approximate importance function.

The third sequence of ten test cases consists of 13 evidence nodes. As done in the former cases the graph is plotted as can be seen in Figure 5.19 between *Hellinger's distance* and $P(E)$. The most likely evidence for 13 evidence node set is 4.87e-07 while the most unlikely evidence captured is 9.32e-54. This value of evidence which is of the order of 10^{-54} is much less as compared to unlikely evidence encountered in [45][49].

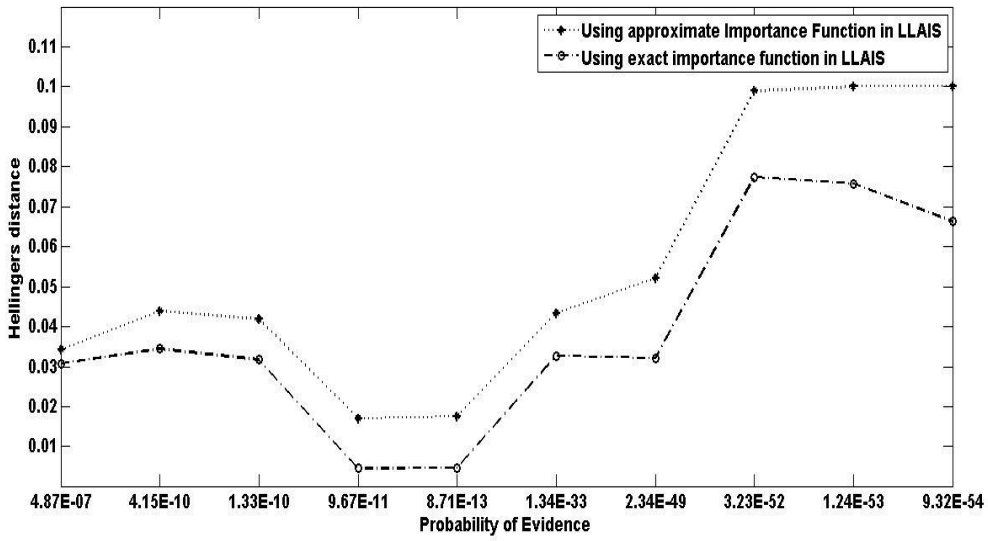

Figure 5.19: Performance comparison of approximate importance function and exact importance function using *Hellinger's distance* on Pathfinder network with 13 evidence nodes.

For the set of 13 evidence nodes as it can be seen in Figure 5.19 that the approximate importance function in LLAIS again did not performed well in comparison to the exact importance function. The sampling output using the exact importance function as we can observe in the graphs above has resulted in relatively less error even in the case when $P(E)$ goes more and more unlikely. The performance of approximate importance function showed large variation as the $P(E)$ was reaching extreme probabilities denoting that LLAIS is not scalable for this network.

From the randomly selected evidence set for this network the most likely event encountered is 3.76e-03 and the most unlikely event being 9.32e-54. The histogram in Figure 5.20 will show the frequency distribution of $P(E)$ to help in analyzing the order of unlikeliness of evidence for this network and it makes us to conclude that from our randomly selected evidence set the most frequently occurring $P(E)$ for this network is of the order of 10^{-7} and 10^{-33}.

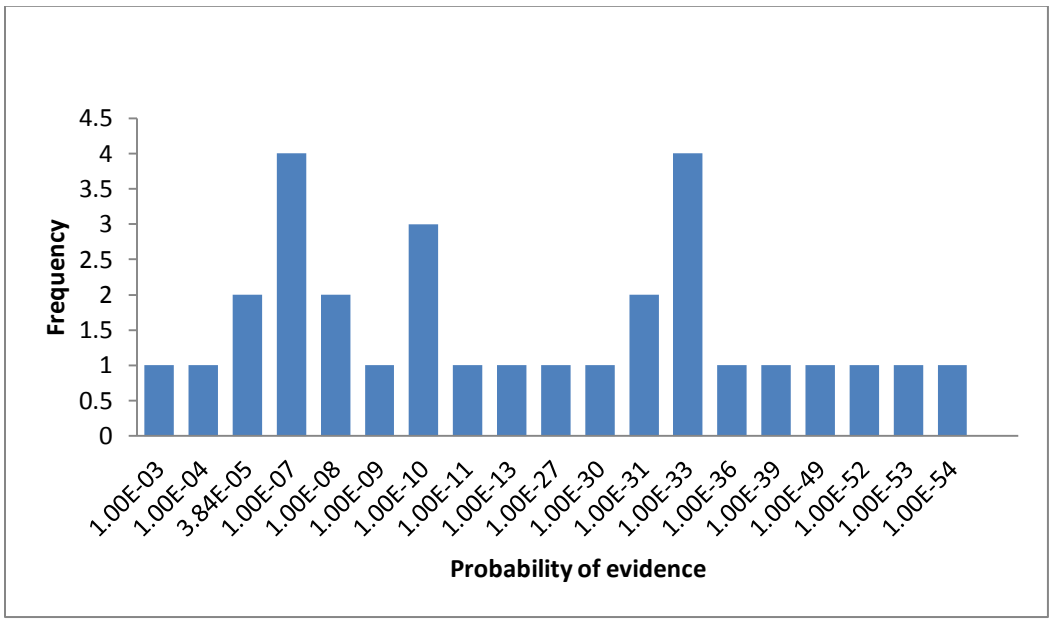

Figure 5.20: Frequency distribution of P(E) in Pathfinder network.

Table 5.5 shows the statistical data of the results obtained from LLAIS using the approximate and exact importance function in terms of *Hellinger's distance*. The minimal *Hellinger's distance* using the approximate importance function in LLAIS is 0.0168 but using the exact importance function it is 0.0038 which is far less than former.

Hellinger's distance	Using Approximate Importance Function	Using Exact Importance Function
Minimum Error	0.0168	0.0038
Maximum Error	0.1	0.0774
Median	0.0379	0.0313
Mean	0.0403	0.0269
Variance	6.05e-04	4.41e-04

Table 5.5: Statistical results for all the 30 test cases generated to test LLAIS for Pathfinder network in terms of *Hellinger's distance*.

On the other hand the maximal *Hellinger's distance* using the exact importance function is 0.0774 which is less than the maximal *Hellinger's distance* 0.1 using approximate importance function in LLAIS. The variance and mean value of *Hellinger's distance* is also less using the exact importance function in contrary to the approximate importance function in LLAIS as can be seen in Table 5.5 above. Figure 5.21 shows the graph for *Hellinger's distance* plotted corresponding to the total 30 test cases generated. We can see in the graph that the exact importance function performs far better than the approximate importance function.

For the comparison of performance measures *MSE* is also calculated as shown in Table 5.6 for all the test cases as we did for the other two networks. The maximal MSE as we

can see in Table 5.6 for approximate importance function in LLAIS is 0.106 while using exact importance function its 0.0038. The variance in the sampling results from approximate importance function is more in comparison to that of the exact importance function, showing that sampling from the approximate importance function of LLAIS results in poor accuracy and it not reliable enough to apply on this network.

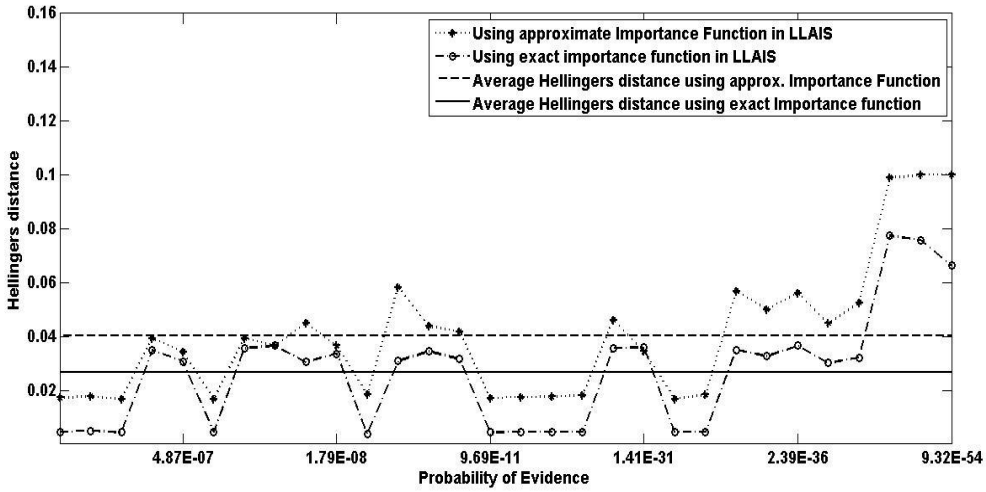

Figure 5.21: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Hellinger's distance* for Pathfinder network.

Mean Square Error	Using Approximate Importance Function	Using Exact Importance Function
Minimum Error	0.0147	0.0038
Maximum Error	0.106	0.0698
Median	0.0389	0.0299
Mean	0.0407	0.0257
Variance	6.31e-04	3.77e-04

Table 5.6: Statistical results for all the 30 test cases generated to test LLAIS for Pathfinder network in terms of *Mean Square Error*.

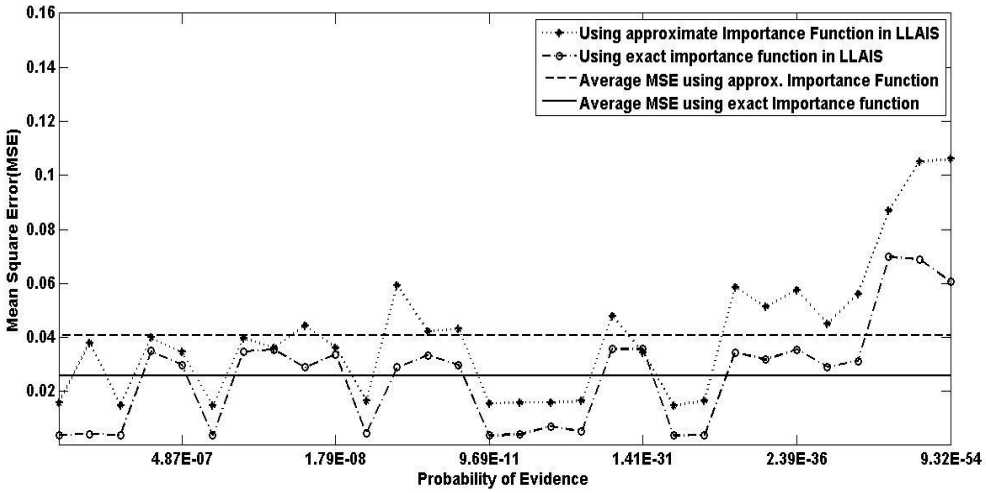

Figure 5.22: Performance comparison of approximate and exact importance function combining all the 30 test cases generated in terms of *Mean Square Error* for Pathfinder network.

The difference in values of statistical measures from the two performance measures is due to the fact *Hellinger's distance* is considered to handle zero probabilities much more accurately than *MSE*.

The sampling from the exact importance function gave significant results in comparison to the approximate importance function on this network and almost took less than 50% of the total time taken by sampling using the approximate importance function in which it updates and learns the optimal distribution. We cannot assert in case of this network that the LLAIS using approximate importance function is scalable and reliable enough as it was in case of Hailfinder network (56 nodes) and Win95pts network (76 nodes) since it performed really bad giving poor precision for Pathfinder network (109 nodes).

5.3 Summary of Testing

In the last section 5.2 the experiment results for testing LLAIS on the larger networks are discussed, and it can be concluded that sampling from the approximate importance function in LLAIS almost gives as good results as given by exact importance function in case of Hailfinder and Win95pts network but its performance degrades once it is applied to the Pathfinder network which is the largest of the three networks being tested.

The reason behind this is there are lots of zero probabilities in case of Pathfinder network and topology of network is quite complex so LLAIS did not showed good performance on this network resulting in large variance of sampling output. During the experiments for testing certain concerns aroused and were fixed.

They are discussed as follows:

- Firstly, many times the exact value of $P(E)$ was encountered to be zero for the test cases generated, especially in case of Pathfinder Network this situation was too frequent. After investigating it is concluded that reason behind it is the fact that the randomly chosen evidences from the network were having extreme values of probabilities as 0 or too close to 0 in original CPTs, hence for the particular assignment of evidence nodes where the evidence nodes were given random values it always resulted in $P(E)$=0. To clarify it further, when the values in original CPTs for the evidence nodes were changed (making them > 0) then for the same set of evidences with same instantiations non zero value of $P(E)$ is yielded.

- Secondly, getting NAN value of error produced for some test cases generated. Since so far LLAIS was tested on 37 nodes network so issue of NAN value did not arise due to absence of extreme probabilities in this small network but once the algorithm was applied to larger network having extreme probabilities NAN was yielded as the sampling output. This problem was also fixed during the testing of LLAIS after dealing with division by zero in the algorithm.

5.4 Tuning the Parameters and Improving LLAIS

The second part of the thesis deals with the tuning of parameters to improve the performance of LLAIS since in the last section 5.2 we have seen that LLAIS performs really bad in case of Pathfinder network. So there is a need to tune these parameters such as threshold, number of updates and updating interval so that the algorithm is able to give good results for all the networks.

The experiments are performed on the same three networks, that is, Hailfinder, Win95pts and Pathfinder; relatively the most extreme probabilities were encountered in Pathfinder and Hailfinder. So we believe that tuning of parameters will result in improvement in performance of sampling. From the empirical testing on the three networks we could not determine any universal value of threshold which can always yield better results, since if one heuristic cutoff works well on one network it did not work on the other. So the nature of distribution of probabilities in the network plays an important role in selection of threshold $\in -cutoff$. After testing and analyzing results for different threshold values, we recommend to use $\in = 0.01$ for nodes with the number of outcomes less than 5,

$\epsilon = 0.006$ for nodes with the number of outcomes between 5 and 8, otherwise

$\epsilon = 0.0005$, the experiments with different cutoff values are motivated from [49].

As discussed earlier in the last chapter 4, we used 4500 samples and used three updates of updating 2100 samples instead of 5000 samples and five updates of 2000 samples. Since we updated the importance function three times instead of five so it resulted in saving time and having different values of threshold resulted in much more accuracy than the original algorithm.

The improved LLAIS has shown quite good results in comparison to the original LLAIS as we will see in the next section 5.5.

5.5 Experiment Results

5.5.1 Experiment Results for Improved LLAIS on Hailfinder BN

The experiments are performed for analyzing the performance of improved LLAIS following the same procedure as followed for the testing of LLAIS as discussed in chapter 4. The graph plotted as shown in gives the comparison of the results from the original LLAIS and improved LLAIS.

The summary of the results from the combination of 30 test cases generated is shown in Table 5.7 below. Table 5.7 describes the comparison of statistical measures for the original LLAIS and the improved LLAIS. The minimal *Hellinger's distance* from improved LLAIS is 0.0076 while for the original LLAIS it is 0.01. The maximal

Hellinger's distance from the improved LLAIS is 0.014 which is far less than 0.0205 obtained from the original LLAIS. Further the average error using the original LLAIS is 0.0128 while for the improved LLAIS it is 0.0101 which is quite less and also the variance in sampling result for improved LLAIS is less in comparison to the original LLAIS.

Hellinger's distance	Original LLAIS	Improved LLAIS
Minimum Error	0.01	0.0076
Maximum Error	0.0205	0.014
Median	0.0119	0.0097
Mean	0.0128	0.0101
Variance	7.08e-06	2.73e-06

Table 5.7: Statistical results for all the 30 test cases generated to compare the performance of original LLAIS with improved LLAIS on Hailfinder network in terms of *Hellinger's distance.*

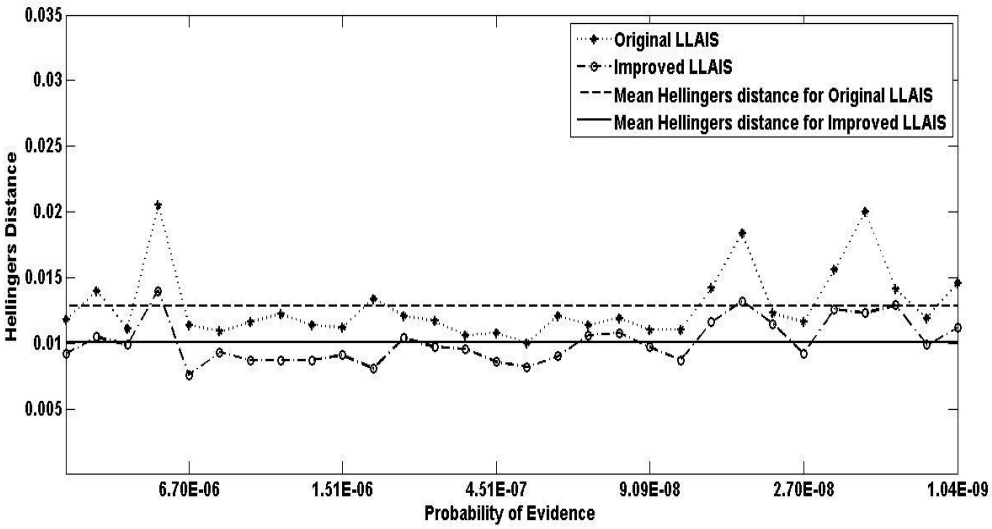

Figure 5.23: Performance comparison of original LLAIS and improved LLAIS for Hailfinder network. *Hellinger's distance* for each of the 30 test cases plotted against P(E).

The graph shown in Figure 5.23 gives the performance of original and improved LLAIS for all the 30 test cases generated where $P(E)$ is taken along x-axis ranging from most likely evidence to most unlikely evidence. The *Hellinger's distance* is plotted along y-axis in the same way as we plotted during the testing of LLAIS.

From the analyses of the graph in Figure 5.23 and Table 5.7 it can be seen that the performance of improved LLAIS after tuning the parameters is quite good in comparison to the original LLAIS. The improved LLAIS is also time effective for it requires less number of samples and less updates than the original one.

Hence for the Hailfinder network we are able to see good performance of improved LLAIS showing better convergence as compared to the original LLAIS.

5.5.2 Experiment Results for Improved LLAIS on Win95pts BN

For this 76 nodes network, the experiments are performed in the same way as done in the earlier cases in Section 5.4.1, that is, generating total of 30 test cases constituting three sequences of ten test cases each of 9, 11 and 13 evidence nodes set respectively.

The following Table 5.8 displays the statistical information for comparing the performance of the original LLAIS and improved LLAIS. The minimal *Hellinger's distance* for the improved LLAIS is 0.0054 while for the original LLAIS it is 0.0087. The maximal *Hellinger's distance* for original LLAIS is 0.02 which is more than 0.0125 given using improved LLAIS. In addition to this, the variance in output is again less in case of

improved LLAIS as compared to the original LLAIS. So for this network too improved LLAIS showed good results by updating the importance function three times only in comparison to original LLAIS.

Hellinger's distance	Original LLAIS	Improved LLAIS
Minimum Error	0.0087	0.0054
Maximum Error	0.02	0.0125
Median	0.0105	0.0075
Mean	0.0114	0.0078
Variance	6.45e-06	2.50e-06

Table 5.8: Statistical results for all the 30 test cases generated to compare the performance of original LLAIS with improved LLAIS on Win95pts network in terms of *Hellinger's distance*.

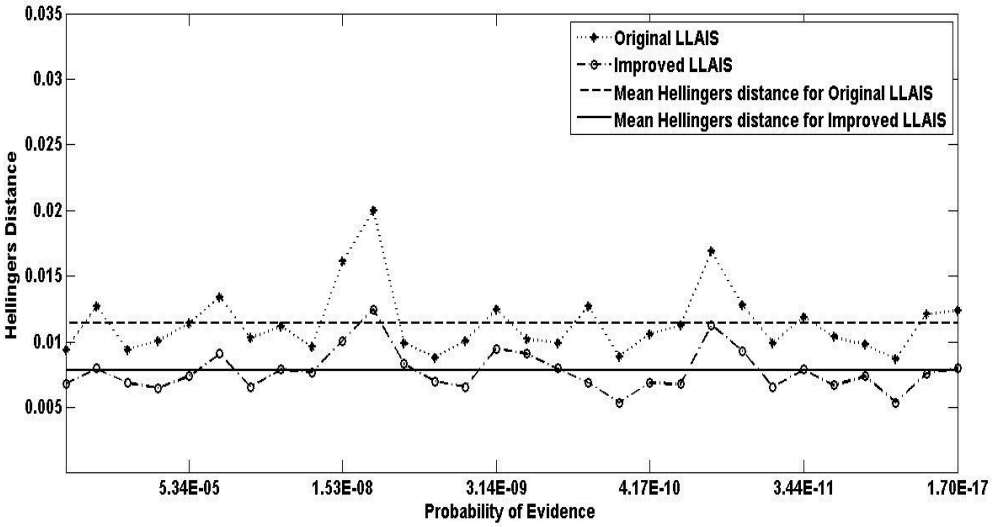

Figure 5.24: Performance comparison of original LLAIS and improved LLAIS for Win95pts network. *Hellinger's distance* for each of the 30 test cases plotted against P(E).

Figure 5.24 shows the graph which is plotted between $P(E)$ and *Hellinger's distance*. The $P(E)$ is arranged along the x-axis in the descending order ranging from the most likely evidence to the most unlikely and the corresponding value of *Hellinger's distance* is plotted on y-axis for all the 30 test cases respectively.

From the graph in Figure 5.24 it can be seen that the performance of improved LLAIS is quite good in comparison to the original LLAIS and also it does not degrade as the $P(E)$ is reaching the extreme value of the order of 10^{-17}.

From the analyses of the graph in Figure 5.24 and Table 5.8, it can be concluded that the performance of improved LLAIS is quite better than the original LLAIS. It is also to be noted that improved LLAIS is also time efficient for it now requires only three updates instead of five updates required by original LLAIS and also gives better performance than latter.

Hence for Win95pts network containing 76 nodes the performance of improved LLAIS is good and shows better convergence in comparison to the original algorithm.

5.5.3 Experiment Results for Improved LLAIS on Pathfinder BN

The Pathfinder network containing 109 nodes has many probabilities as 1 and 0 showing deterministic relationship between the nodes. It contains many extreme probabilities that have to be dealt with the heuristic cutoff so that the algorithm will result in good precision. So, following the same experiment methods as done in the previous cases 30

test cases were for this network consisting of three sequences containing 10 test cases of 9, 11 and 13 evidence nodes set respectively. As seen in the last section 5.2, LLAIS performed very poorly on this large network and has shown large variance in the result so there is a need to improve it and the modification done by the tuning of parameters proved to be very effective in reducing the error rate for Pathfinder network in comparison to the original LLAIS as can be seen in the Table 5.9.

Hellinger's distance	Original LLAIS	Improved LLAIS
Minimum Error	0.0168	0.0068
Maximum Error	0.117	0.0451
Median	0.0387	0.0149
Mean	0.0427	0.0166
Variance	7.80e-04	1.09e-04

Table 5.9: Statistical results for all the 30 test cases generated to compare the performance of original LLAIS with improved LLAIS on Pathfinder network in terms of *Hellinger's distance*.

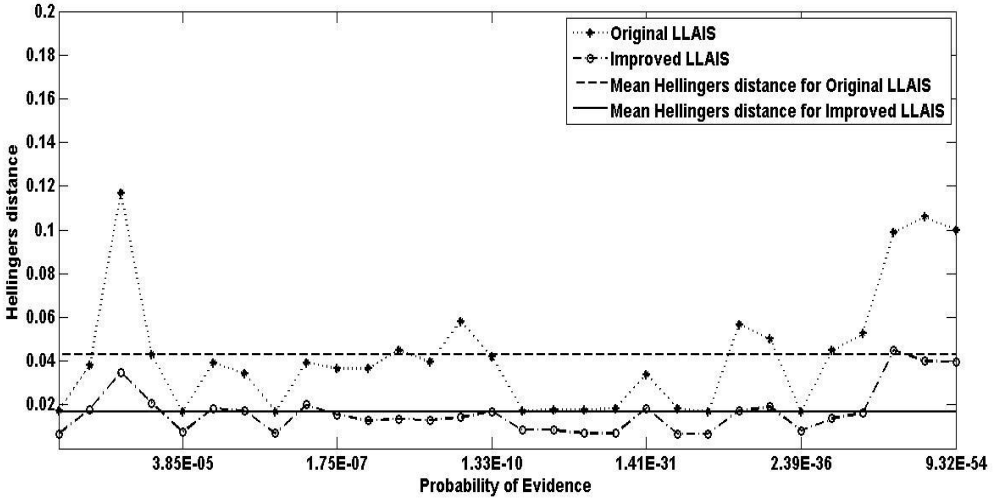

Figure 5.25: Performance comparison of original LLAIS and improved LLAIS for Pathfinder network. *Hellinger's distance* for each of the 30 test cases plotted against P(E).

As can be seen in the table below the maximal *Hellinger's distance* for original LLAIS is 0.117 while for the improved it is 0.0591 which is far less. The minimal *Hellinger's distance* computed from the improved LLAIS is 0.0068 while for the original LLAIS is 0.0168. The variance in case of improved LLAIS is quite less which again shows that improved LLAIS is better in comparison to the original LLAIS.

The graph is plotted as can be seen in Figure 5.24 for all the 30 test cases generated. It can be observed that the improved LLAIS shows good performance and even for the unlikely evidence it did not performed as bad as original LLAIS resulting in the maximum average error which is almost 50% of that of the error from the original LLAIS.

Hence it can be concluded that improved LLAIS is effective in time requiring less updates and less samples as well as more accurate than the original LLAIS.

5.6 Summarizing the Experiments in terms of Time Taken

As we have seen in chapter 3, LLAIS used 5000 samples, 5 updates and updating interval of 2000 to reach the optimal distribution and give results. For testing LLAIS on larger networks, the exact importance function is calculated so that it is easy to determine how close the approximate importance function is able to reach optimal importance function.

The framework in [4] gave us an opportunity to compute the exact value of importance function which does not require learning and updating and hence saving a lot of time. The following Table 5.10 displays the comparison of time taken by the approximate and exact

importance function in LLAIS. The time shown in Table 5.10 includes the time taken by the approximate importance function when it is initialized, updated using the previous set of samples five times, learned and producing the posteriors. The time displayed is the average of time taken by the algorithm for ten test cases.

As we can see in Table 5.10 a lot of time is saved when exact importance function is used since it does not require updating and learning process while the approximate importance function takes too much time especially in case of Pathfinder network where the single iteration is too time consuming.

Name of Network	Average time taken using Approximate Importance Function(LLAIS) in *minutes*	Average time taken using Exact importance Function in *minutes*
Hailfinder	8.742	2.362
Win95pts	10.328	3.482
Pathfinder	30.751	8.203

Table 5.10: Comparison of the average time taken for ten test cases by approximate importance function and exact importance function in producing the posterior probabilities.

Furthermore, the tuning of parameters apart from improving the performance of LLAIS in terms of accuracy also resulted in saving of time, though not so significant difference in consumption of time is recorded but still taking less time in comparison to the original LLAIS and also giving better precision.

From the experiments done for testing the performance of improved LLAIS as seen in section 5.5 it is seen that the estimation of posterior beliefs for non-evidence nodes from improved LLAIS is quite better as compared to the original LLAIS. The data in Table 5.11 shows the difference in time taken by improved LLAIS and original LLAIS taking the average time taken by ten test cases; So we can compare that for every sequence of 10 test cases, improved LLAIS takes around 33 minutes less for 56 nodes network, 28 minutes less for 76 nodes network and approximately 95 minutes less for 109 nodes network; showing that the improved LLAIS is quite time efficient as compared to the original LLAIS.

Name of Network	Average time taken using original LLAIS in *minutes*	Average time taken using improved LLAIS in *minutes*
Hailfinder	9.087	6.032
Win95pts	9.502	6.701
Pathfinder	33.905	22.091

Table 5.11: Comparison of the average time taken for ten test cases by original LLAIS and improved LLAIS in producing the posterior probabilities.

It can be concluded that calculating the exact importance function resulted in saving a lot of time since no time is spent on updating and learning of optimal importance function. On the other the tuning of various tunable parameters not only resulted in improving the accuracy of the original algorithm as seen in the last section 5.5 but also made it time efficient to some extent.

5.7 Discussion

In this chapter we have presented and discussed the experiment results. The approach and procedure for testing LLAIS by comparing results from the exact importance function and improving the algorithm LLAIS by tuning the parameters are briefly discussed. We used three networks for the experimentation - Hailfinder, Win95pts and Pathfinder consisting of 56, 76 and 109 nodes respectively; also these networks are treated as subnets or local JT in MSBN. It is to be noted that the application of this algorithm to such big subnets is not been reported yet.

From the experiments performed for testing of LLAIS on large networks we can conclude that the performance of the approximate importance function in LLAIS for estimating the posterior probabilities in case of Hailfinder and Win95pts networks is quite good when compared to the exact importance function but once the algorithm is applied to Pathfinder (109 nodes) it did not gave good results. So the algorithm was not scalable and reliable enough when applied to large network which denote in itself local JT in LJF.

The proper tuning of parameters resulted in effective results by improving the algorithm. It was seen that adjustment of small probabilities led to the significant betterment in the performance of algorithm and now it requires less number of samples and less updates to for estimating posteriors. So we can say that if the network contains extreme probabilities then for good precision adjustment of these tunable parameters will lead to convincing and sound results as seen in the experiment results in this chapter.

Chapter 6

Conclusion

In this chapter thesis summary will be discussed and then some additional remarks will be given for future research.

6.1 Thesis Summary

Since the exact inference algorithms have been proved to be NP-hard [3], so it led to the development of various approximate inference algorithms so that they can be applied to decision theory in knowledge-based systems. In MSBN where a large BN is sectioned into sub-domains it becomes very important to deal with the reasoning between the sub-domains so formed, in addition to this, as the size of those sub-domains increases and become complex, the exact inference becomes quite difficult and costly. For example, [4] network may contain subnets that are too large and complex to allow for the exact local representation. So it is obvious to trade off exact inference against the calculation speed and communication cost with approximate approaches [4].

In MSBN, LJF provides a coherent framework for doing inference in MSBN. LLAIS developed by [4] is the extension of BN importance sampling techniques to JTs and it integrates local sampling with existing LJF framework. The prototype of LLAIS was so far tested on 37 nodes Alarm network and it was important to test the scalability and reliability of this algorithm on larger network and as we know that the best way to test the

approximate inference algorithms is to apply them on larger networks.

For this purpose we chose three networks named –

(i) Hailfinder (56 nodes) (ii) Win95pts (76 nodes) and (iii) Pathfinder (109 nodes).

We limited ourselves up to 109 nodes network since LLAIS is local adaptive sampling and has to be applied to the local JT which is formed after sectioning of large BN. So testing LLAIS up to 109 nodes network is a quite adequate decision.

The testing of LLAIS is done by comparing the performance of sampling from the approximate importance function in LLAIS with that from the exact importance function. The main idea is to test the algorithm on larger network for its performance when the $P(E)$ goes unlikely. The departure from the exact solution is computed using variable-elimination algorithm. The accuracy is preferred to be measured in terms of *Hellinger's distance* rather than *Mean Square Error* since former weights small absolute probability differences near 0 much more heavily than similar probability differences near 1.

The networks for testing were downloaded from Genie and Smile Bayesian repository and platform used for experiments is MATLAB. From the testing of LLAIS on large networks we concluded that the algorithm performed quite well in case of Hailfinder and Win95pts network but did not showed good results in case of Pathfinder network. The reason behind it is the presence of extreme probabilities in Pathfinder that have to adjusted to produce better results.

The second part of thesis is to improve LLAIS by tuning the various tunable parameters such as threshold value of $\in -cutoff$, number of samples, updating interval and number of updates. Since some of the Bayesian networks contain many small probabilities so they need to be dealt properly for getting better precision. In this regard different threshold values were adjusted for nodes with different number of outcomes. It resulted in improving the performance of algorithm with less number of updates and less number of samples.

From the empirical testing of LLAIS, tuning the parameters resulted in improving the algorithm. The setting of updating interval to 2100 required only three updates instead of five (as used by original algorithm) to get much better results with less number of samples in comparison to the original LLAIS and hence making it more time efficient. The tuning of parameters resulted in improving the accuracy of the original algorithm on all the three networks.

6.2 Future Work

It has been seen that learning time of the optimal importance function takes too long, so the choice of initial importance function $Pr^0(X\backslash E)$ close to the optimal importance function can greatly affect the accuracy and convergence in the algorithm. The algorithm can be improved by developing methods for estimating the posterior distribution with better accuracy. As mentioned in [4], there is still one important question that remains unanswered how the local accuracy will affect the overall performance of the entire network. Further experiments are still to be done on the full scale MSBNs.

Bibliography

[1] Karen H. Jin, "Efficient probabilistic inference algorithms for cooperative multi-agent systems", Ph.D. dissertation, University of Windsor (Canada), Canada. Retrieved from: Dissertations & Theses @ University of Windsor. (Publication No. AAT NR71662).
[Date accessed]: 25 October, 2011.

[2] Y.Xiang, "Comparison of multiagent inference methods in multiply sectioned Bayesian networks", *International journal of approximate reasoning*, vol. 33, pp.235-254, 2003.

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

[4] Karen H. Jin and Dan Wu, "Local Importance Sampling in Multiply Sectioned Bayesian Networks", *Florida Artificial Intelligence Research Society Conference*, North America, May. 2010. Available at:
http://www.aaai.org/ocs/index.php/FLAIRS/2010/paper/view/1315.
[Date accessed]: 14Jan, 2012.

[5] Decision System Laboratories, About Genie and Smile. University of Pittsburgh, 2008. Available at: http://genie.sis.pitt.edu/about.html,
[Date accessed]: 19th August'2012.

[6] Daphne Koller and Nir Friedman, Probabilistic Graphical Models-Principles and Techniques, MIT Press, 2009.

[7] Jian Cheng, "Efficient Stochastic Sampling Algorithms for Bayesian networks", Ph.D. Dissertation, School of Information Sciences, University of Pittsburgh, 2001. Available at: http://www2.sis.pitt.edu/~jcheng/Cheng_dissertation.pdf.
[Date accessed]: 22September, 2011.

[8] Y.Xiang, Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach". Cambridge University Press, 2002.

[9] G.Shafer, Probabilistic Expert Systems. Society for Industrial and Applied Mathematics, 1996.

[10] H.Guo and W.Hsu, A survey on algorithms for real-time Bayesian Network Inference, In Proceedings of the Joint AAAI-02\KDD-02\UAI-02 workshop on Real-time decision support and diagnosis systems, Edmonton, Canada, 2002.

[11] R.G.Cowell, A.P.Dawid, S.L.Lauritzen and D.J.Spiegelhalter, "Probabilistic Networks and Expert Systems", Springer, 1999.

[12] M.Yannakakis, "Computing the minimum fill-in is NP-complete", *SIAM Journal of Algebraic and Discrete Mathematics,* vol.2, pp.77-79, 1981.

[13] J.Blair, P.Heggernes and J.Telle, "A practical algorithm for making filled graphs minimal", *Theoretical Computer Science,* vol.250, pp.125-141, 2001.

[14] E.Dahlhaus, "Minimal elimination ordering inside a given chordal graph", *in Graph-Theoretic Concepts in Computer Science,* Springer, 1997, pp.132-143.

[15] F.V.Jensen and Frank Jensen, Optimal Junction trees. In *Proceedings of the 10th conference on uncertainty in Artificial Intelligence*, pp.360-366, San Francisco, CA, USA, 1994.

[16] F.V.Jensen, S.L.Lauritzen and K.G.Olesen, "Bayesian updating in causal probabilistic networks by local computation", *Computational statistics Quarterly*, vol.4, pp.269-282, 1990.

[17] A.Madsen, F.Jensen, U.B.Kjaerulff and M.Lang, "The hugin tool for probabilistic graphical models", *International Journal on AI tools,* vol.14, no.3, pp.507-543, 2005.

[18] P.P. Shenoy and G.Shafer, "Propagating belief functions using local computation", *IEEE Expert*, vol.1, no.3, pp.43-52, 1986.

[19] P.P.Shenoy and G.Shafer, "An axiomatic framework for Bayesian and belief function propagation", In *fourth Conference on uncertainty in Artificial Intelligence,* pp.307-314, St.Paul, MN, 1988.

[20] P.P.Shenoy, "Binary join trees for computing marginal in the Shenoy-Shafer architecture", *International Journal of Approximate Reasoning,* vol.17, no.1, pp.1-25, 1997.

[21] P.Dagum and M.Luby, "Approximating probabilistic Inference in Bayesian Belief Networks is NP-hard", *Artificial Intelligence,* vol.60, no.1, pp.141-153, 1993.

[22] A.Kozlov and J.Singh, "Computational complexity reduction for BN20 networks using similarity of states", *In Proceeding of the Twelfth Annual conference on Uncertainty in Artificial Intelligence (UAI-96),* pp.357-364, San Francisco, 1996.

[23] U. Kjaerulff, "Reduction of computational complexity in Bayesian networks through removal of weak dependencies", In *tenth conference on uncertainty in Artificial Intelligence*, pp.374-382, 1994.

[24] M.Henrion, "Search-based methods to bound diagnostic probabilities in very large belief nets", In *Proceedings of seventh conference on uncertainty in Artificial Intelligence*, 1991.

[25] D.Poole, "The use of conflicts in searching Bayesian Networks", In *Proceedings of the ninth conference on Uncertainty in Artificial Intelligence in Artificial Intelligence*, pp.359-367, Washington D.C, 1993.

[26] D.Poole, "Average-case analysis of a search algorithm for estimating prior and posterior probabilities in Bayesian Networks with extreme probabilities", In *Proceeding of thirteenth International Joint Conference on Artificial Intelligence*, pp.606-612, France, 1993.

[27] D.Poole, "Probabilistic conflicts in a search algorithms for estimating posterior probabilities in Bayesian Networks", *Artificial Intelligence,* vol.88, no.1-2, pp.69-100, 1996.

[28] K.P.Murphy, Y.Weiss and M.Jordan, "Loopy belief propagation for approximate inference: an empirical study", In *the Proceedings of uncertainty in Artificial Intelligence*, pp.467-475, 1999.

[29] R.J.McEliece, D.J.MacKay and J.F.Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm", *IEEE journal of selected areas of communication*, pp.140-152, 1998.

[30] K.P.Sycara, Multiagent systems, *AI Magazine*, vol.19, no.2, pp.79-92, 1998.

[31] M.Wooldridge and N.R.Jennings, "Intelligent agents: theory and practice", *Knowledge Engineering Rev*iew, vol.10, no.2, pp.115-152, 1995.

[32] Y. Xiang and V. Lesser, "Justifying multiply sectioned Bayesian networks*",* In *Proceedings of the fourth International Conference on Multi-agents Systems*, pp.349-356, Boston, 2000.

[33] A.Ghosh and S.Sen., "Agent-based distributed intrusion alert system", *In the sixth International workshop on Distributed Computing-IWDC*, pp.7-47, Kolkatta, India, 2005.

[34] Y.Xiang and H.Geng, "Distributed monitoring and diagnosis with multiply sectioned Bayesian networks", *In AAAI Spring symposium on Artificial Intelligence in Equipment service Maintenance and Support*, pp.18-25, 1999.

[35] Y.Xiang, B.Pant, A.Eisen, M.P.Beddoes and D.Poole, "Multiply Sectioned Bayesian Networks for neuromuscular diagnosis", *Artificial Intelligence in Medicine*, vol.5, no.4, pp.293-314, 1993.

[36] D.Koller and A.Pfeffer, "Object Oriented Bayesian networks", In *thirteenth conference on uncertainty in Artificial Intelligence*, pp.302-313, Morgan Kauffmann Publishers, 1997.

[37] K.H.Jin and D.Wu, "Marginal calibration in multi-agent probabilistic systems", In *Proceedings of the 20th IEEE International conference on Tools with AI*, 2008.

[38] K.H.Jin, Dan Wu and Libing Wu, "On designing approximate inference algorithms for multiply sectioned Bayesian networks", *IEEE International Conference on Granular Computing, 2009, GRC'09*, pp. 294-299, 2009.

[39] Y.Xiang, "A probabilistic framework for cooperative multi-agent distributed interpretation and optimization for communication", *Artificial Intelligence*, vol.87, pp.295-342, 1996.

[40] Y.Xiang, "Belief updating in multiply sectioned Bayesian networks without repeated local propagations", *International Journal of Approximate Reasoning*, vol.23, pp.1-21, 2000.

[41] Yang Xiang, F.V.Jensen and Xiaoyun Chen, "Inference in multiply sectioned Bayesian networks: methods and performance comparison", *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol.36, no.3, pp.546-558, June 2005.

[42] Y.Xiang, D.Poole and M.P.Beddoes, "Multiply sectioned Bayesian networks and junction forests for large knowledge based systems", *Computational Intelligence*, vol.9, no.2, pp.171-220, 1993.

[43] R. Fung and K. C. Chang, "Weighting and integrating evidence for stochastic simulation in Bayesian networks", In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, pp.209–220, 1990.

[44] R. Shachter and M. Peot, "Simulation approaches to general probabilistic inference on belief networks", In *Fifth Conference on Uncertainty in Artificial Intelligence*, pp.221–231, 1990.

[45] J. Cheng and M. J. Druzdzel, " BN-AIS: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks", *Artificial Intelligence Research*, vol.13, pp.155–188, 2000.

[46] M.A.Shwe and G.F.Cooper, "An empirical analysis of likelihood-weighting simulation on a large, multiply-connected medical belief network", *Computers and Biomedical Research*, vol.24, no.5, pp.453-475, 1991.

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

[48] C. Yuan and M. J. Druzdzel, "An Importance sampling algorithm based on evidence pre-propagation", In *19th Conference on Uncertainty in Artificial Intelligence*, pp.624–631, 2003.

[49] C. Yuan, “Importance Sampling for Bayesian Networks: Principles, Algorithms, and Performance”, PhD thesis, University of Pittsburgh, 2006.

[50] D. MacKay, “Introduction to Monte Carlo methods”, In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models,* pp.175–204, 1998.

[51] U. Kjaerulff, “Hugs: Combining exact inference and Gibbs sampling in junction Trees”, In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 368–375, 1995.

[52] M. Paskin, Sample propagation. In S. Thrun, L. Saul, and B.Scholkoph, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, MIT Press, 2004.

[53] S. Moral, L. D. Hernndez and A. Salmern, “A Monte Carlo algorithm for probabilistic propagation in belief networks based on importance sampling and stratified simulation techniques”, *International Journal of Approximate Reasoning*, vol.18, pp.53–91, 1998.

[54] B.Abramson, J.Brown, W.Edwards, A.Murphy and RL.Winkler, "Hailfinder: A Bayesian system for forecasting severe weather", *International Journal of Forecasting*, vol.12, no.1, pp.57-71, 1996.

[55] David E. Heckerman, E. J. Horvitz and B. N. Nathwani, “Toward Normative Expert Systems: Part I the Pathfinder Project.”, In *Methods of Information in Medicine*, vol.31, pp.90-105, 1992.

[56] Norsys Software Corp. Available at: http://www.norsys.com/netlibrary/index.htm [Date accessed]: 19th August’2012.

Vita Auctoris

NAME	Sonia Bhatti
PLACE OF BIRTH	Punjab, India
YEAR OF BIRTH	1985
EDUCATION	Department of Computer Science Dr. B. R. Ambedkar National Institute of Technology Jalandhar, Punjab, India B.E (2003 - 2007)
	School of Computer Science University of Windsor, Windsor, Ontario, Canada M.Sc. (2010 - 2013)