Knowledge representation, communication, and update in probability-based multiagent systems

By

Scott Langevin

Bachelor of Science
University of Victoria 2001

Master of Engineering
University of South Carolina 2008

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science and Engineering
College of Engineering and Computing
University of South Carolina

2011

Accepted by:
Marco Valtorta, Major Professor
Michael N. Huhns, Chairman, Examining Committee
Manton Matthews, Committee Member
José M. Vidal, Committee Member
John Byrnes, External Examiner
Tim Mousseau, Interim Dean of the Graduate School
UMI Number: 3454755

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.

UMI 3454755
Copyright 2011 by ProQuest LLC.
All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.
© Copyright by Scott Langevin, 2011  
All Rights Reserved.
ACKNOWLEDGMENTS

I would like to thank my dissertation committee for providing me the guidance needed to complete my dissertation. Special thanks to my advisor, Dr. Marco Valtorta. I would never have succeeded without your guidance. You helped navigate me through the murky and rocky waters of completing a dissertation. Your passion and depth of knowledge is an inspiration to me. I will forever be grateful to you and cherish the time we spent together. To Dr. Michael N. Huhns, I am very grateful for all you have done for me. You took me under your wing and ensured I was funded throughout my degree. I have learned so much from you. You are a great mentor and teacher. To Dr. Manton Matthews, I hope I will always be your “favorite Canadian”. You are my favorite graduate director. Thanks to Dr. José Vidal and Dr. John Byrnes for being on my committee and for the excellent feedback and suggestions.

To my fur kids, Wraith and Onyx, you were always there to cheer me up and keep me sane. Finally, thanks to my best friend and wife, Robbi, I would never would have returned to academia without your encouragement. Thank you for all your support and patience. Your love gave me the fuel to carry on.
Abstract

In this dissertation, we define a cooperative multiagent system where the agents use locally designed Bayesian networks to represent their knowledge. Agents communicate via message passing where the messages are beliefs in shared variables that are represented as probability distributions. Messages are treated as soft evidence in the receiver agents, where the belief in the receiving agent is replaced by the publishing agent’s belief. We call this the oracular assumption, where one agent is an expert or more knowledgeable of particular variables. As a result, the agents are organized in a publisher-subscriber hierarchy. A central problem of message passing in probabilistic systems is the so called rumor problem, where cycles in message passing cause redundant influence of beliefs. We develop algorithms to identify and solve the rumor problem in the context of our multiagent system. We compare and contrast our system with the MSBN multiagent model.

Central to our agent model is the notion of soft evidential update. We develop methods to efficiently perform probabilistic update in Bayesian networks where the soft evidence is respected. We analyze the theoretical and experimental complexity of our methods and compare them with other methods that have been proposed.

Finally, we implement several multiagent systems for experimentation using our multiagent system and MSBNs. We devise performance measures to compare the two systems. From this comparison, we provide guidance for the design of probabilistic multiagent systems.
# CONTENTS

Acknowledgments ................................................................. iii

Abstract ................................................................................ iv

List of Tables ........................................................................ vii

List of Figures ......................................................................... viii

Chapter 1 Introduction ............................................................ 1
  1.1 Research Goal ................................................................. 2
  1.2 Organization of the Dissertation ........................................ 3

Chapter 2 Background ............................................................. 4
  2.1 Agents and Multiagent systems ........................................ 4
  2.2 Uncertain Reasoning ....................................................... 7
  2.3 Related Work .................................................................. 26

Chapter 3 Probabilistic Multiagent Systems .............................. 32
  3.1 Multiply Sectioned Bayesian Networks ............................. 32
  3.2 Agent Encapsulated Bayesian Networks ............................ 37

Chapter 4 The Rumor Problem ................................................ 46
  4.1 Redundant Influences .................................................... 46
  4.2 Identifying Redundant Influences .................................... 49
  4.3 Communication Solution ............................................... 54
  4.4 Coherent AEBSN Systems .............................................. 61
List of Tables

Table 5.1 Complexity of soft evidential variants ........................................ 86
Table 5.2 Statistics for test networks .......................................................... 87
Table 6.1 Bio-attack sequence of events ...................................................... 95
Table 6.2 Evidence phases for Bio-attack simulation. .................................. 102
Table 6.3 Conditional probability table for discount variable. ....................... 103
Table 6.4 Beliefs of Coordinated Bio Attack over scenario phases. ............... 106
Table 6.5 Beliefs of Local Bio Attack over scenario phases. ......................... 106
Table 6.6 Beliefs of Non-Bio Attack over scenario phases. ........................ 107
Table 6.7 Beliefs of No Attack over scenario phases. ................................. 107
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Conceptual agent model as presented in [53].</td>
</tr>
<tr>
<td>2.2</td>
<td>Cold or Angina? Bayesian network.</td>
</tr>
<tr>
<td>2.3</td>
<td>d-separation connection types.</td>
</tr>
<tr>
<td>2.4</td>
<td>The triangulated moral graph (a) and junction tree (b) for the Cold or Angina? example.</td>
</tr>
<tr>
<td>2.5</td>
<td>The Kalman filter represented as a hidden Markov model for a linear dynamic system with internal state $X_t$, and sensor observation $Z_t$.</td>
</tr>
<tr>
<td>3.1</td>
<td>Loss of dependence between variable A and B.</td>
</tr>
<tr>
<td>3.2</td>
<td>Introduction of an observation variable in a subscriber agent for absorption of a publisher message over shared variables $I_1, ..., I_k$.</td>
</tr>
<tr>
<td>3.3</td>
<td>Redundantly Observed Sensor Example (ROSE) communication graph.</td>
</tr>
<tr>
<td>4.1</td>
<td>Example of a rumor initiated by the dinner party host, labeled “Start”, spreading around a dining room table. The rumor reaches the guest labeled “End” from two distinct neighbors, doubly influencing them.</td>
</tr>
<tr>
<td>4.2</td>
<td>$n$ node disjoint paths between $V_i$ and $V_j$.</td>
</tr>
<tr>
<td>4.3</td>
<td>Extra redundant influences that are not node-disjoint (four cases).</td>
</tr>
<tr>
<td>4.4</td>
<td>The ROSE redundancy graph.</td>
</tr>
<tr>
<td>4.5</td>
<td>Multiple overlapping redundant influences.</td>
</tr>
<tr>
<td>4.6</td>
<td>The extended ROSE redundancy graph.</td>
</tr>
</tbody>
</table>
Figure 4.7  The communication graph (a) and resulting redundancy graph (b) illustrating the coherency problem. .............................. 61

Figure 4.8  Example illustrating strongly and weakly coherent systems. If A, B, C and D are locally coherent then the system is strongly coherent. If only C and D are locally incoherent, then the system is weakly coherent. .................................................. 64

Figure 4.9  Graphical structure that leads to incoherence is shown in figure a), if there exists a directed path that includes all the nodes in A or all the nodes in B then the graph is coherent as shown in figure b) and c). .................................................. 66

Figure 4.10  Design solutions for example in Figure 4.7(a) that ensure global coherence. ................................................................. 68

Figure 4.11  Runtime solution for example in Figure 4.7(a). Proxy agent F is introduced to reconcile conflict between agents C and D over P(α,β). The grey arrows are communication links that are removed in the original communication graph, and dashed lines are new communication edges introduced. ...................... 70

Figure 4.12  Messages received by agent a_k from agents a_i and a_j. ...... 72

Figure 4.13  Agent a_j is an ancestor of agent a_i. The dotted line represents a directed path of any length (number of edges) greater than zero. 72

Figure 4.14  Agent a_i and a_j share a common ancestor a_s. The dotted lines represent a directed path of any length (number of edges) greater than zero. .................................................. 72

Figure 5.1  Average number of elementary table operations for the alarm network. ................................................................. 88

Figure 5.2  Average propagation time for the alarm network. .............. 88
Figure 5.3  Number of elementary table operations for test case 3 of the
alarm network. ......................................................... 89
Figure 5.4  Average number of elementary table operations for the test61
network. ................................................................. 89
Figure 5.5  Number of elementary table operations for test 6 of the test61
network. ................................................................. 90
Figure 5.6  Average number of elementary table operations for the test71
network. ................................................................. 90
Figure 5.7  Average number of elementary table operations for the stud
farm network. ......................................................... 91

Figure 6.1  Bio-attack full Bayesian network. ............................. 96
Figure 6.2  Communication graph for bio-attack AEBN simulation (only
Chicago and Kansas agents shown). .................................. 97
Figure 6.3  Bayesian network for Incident agent. ......................... 100
Figure 6.4  Bayesian network for Chicago Human Attack Type agent. . 100
Figure 6.5  Bayesian network for Chicago Human Early Indicator agent. . 101
Figure 6.6  Bayesian network for Chicago Livestock Attack Type agent. . 101
Figure 6.7  Bayesian network for Chicago Livestock Early Indicator agent. 101
Figure 6.8  Discounting external evidence given an accurate test modeling


technique. ................................................................. 104
Figure 6.9  Revised Bayesian network for Chicago Human Attack type agent. 104
Figure 6.10 Revised Bayesian network for Chicago Livestock Attack type agent. 104
Figure 6.11 Beliefs of Coordinated Bio Attack over scenario phases. ........ 108
Figure 6.12 Beliefs of Local Bio Attack over scenario phases. ............... 108
Figure 6.13 Beliefs of Non-Bio Attack over scenario phases. ............... 109
Figure 6.14 Beliefs of No Attack over scenario phases. ....................... 109
Figure 6.15  CD and I-divergence of BioAttackType distribution in AEBN v2 and MSBN over scenario phases. 110

Figure 6.16 Total communication cost of AEBN v2 and MSBN over scenario phases. 111

Figure 6.17 Theoretical scalability of AEBN. 112

Figure 6.18 Scalability of AEBN as number of agents and edges increases. 113

Figure 6.19 Communication graph for expanded bio-attack AEBN simulation. 115

Figure 6.20 Linked Junction Forest for bio-attack MSBN simulation (only Chicago and Kansas agents shown). 116
Chapter 1

Introduction

Large real world intelligent systems are often too complex or expensive to build as centralized systems. The computational cost of the large scale reasoning required can be too prohibitive, the scale and scope of the system too complex for a monolithic system, as well aspects of the system are often distributed physically further complicating the construction of a single agent system. To overcome these challenges, the inference and decision making tasks can be decomposed into sub-problems that are reasoned about locally. Multiagent systems can be used to achieve this modular system design, where each agent is responsible for one or more sub-problems. Through agent communication information is exchanged between agents in order to achieve distributed inferencing and decision making.

The reasoning and decision making task often must cope with uncertainty in the problem domain. The uncertainty may come from unobservable aspects of the domain that must be estimated from aspects that are observable, incomplete understanding of the domain, observations that are imprecise, ambiguous, noisy, or unreliable, and lack of resources necessary to observe all relevant events.

Bayesian networks are a probabilistic framework for reasoning with uncertainty. Although Bayesian networks have greatly reduced the time, space and design complexity involved with reasoning using a probability distribution, large complex single networks are challenging to design. Often the computational cost of exact inference is not possible and approximate methods must be employed. To overcome these limitations, what is needed is to divide the network into smaller, manageable units that
are locally reasoned upon and aggregated to solve the global problem. Often this task is described as distributed Bayesian networks.

In this dissertation, we provide clear assumptions about agents that use probabilistic representations of knowledge, guidelines for their design, and efficient algorithms for communicating (or sharing) probabilities. The goal is to allow easier design of probability-based agents and multiagent systems, resulting in rational decision making. Previous approaches to this problem have imposed strong restrictions on the topology of agent communication, tightly coupled the agents, and have not emphasized the autonomy of each agent. Our agent model attempts to address these deficiencies by loosely coupling the agents and allow for more flexible agent topologies.

1.1 Research Goal

Our goal in this dissertation is the design of a cooperative multiagent system where the agents use locally designed Bayesian networks to represent their knowledge. In this system, agents communicate via message passing where the messages are beliefs in shared variables that are represented as probability distributions. Messages are treated as soft evidence in the receiver agents, where the belief in the receiving agent is replaced by the publishing agent’s belief. We call this the oracular assumption, where one agent is an expert or more knowledgeable of particular variables. As a result, the agents are organized in a publisher-subscriber hierarchy. Our model extends the Agent Encapsulated Bayesian Network (AEBN) system proposed by Bloemeke [3]. This dissertation aims to expand and correct technical details, as well as provide a theoretical basis for AEBNs.

A central problem of message passing in probabilistic systems is the so called rumor problem, where cycles in message passing cause redundant influence of beliefs. We show the solution proposed by Bloemeke is deficient, and develop new algorithms to identify and solve the rumor problem in the context of our multiagent system.
A comparison with other frameworks is presented to highlight the advantages and disadvantages of each, as well as argue for the role of AEBNs in designing probabilistic multiagent systems.

Central to our multiagent system is the notion of soft evidential update. We develop methods to efficiently perform probabilistic update in Bayesian networks where the soft evidence is respected. We analyze the theoretical and experimental complexity of our methods and compare them with other methods that have been proposed.

Finally, we plan to implement a multiagent system for experimentation using our multiagent system formalism and compare implementations of the same examples using Xiang’s MSBN formalism. The evaluation will require the formulation of meaningful performance measures to compare the two systems.

We hypothesize that message passing in probabilistic multiagent systems that are organized in a DAG topology can solve the rumor problem under some restrictive assumptions.

1.2 Organization of the Dissertation

The remainder of this dissertation is organized as follows: Chapter 2 provides relevant background and related work necessary to understand the theoretical basis and context of our research. Chapter 3 presents closely related research and introduces our agent model. Chapters 4 and 5 present the rumor problem in probabilistic systems and soft evidential update respectively. Chapter 6 presents a simulation devised to evaluate our proposed system and scalability analysis. Finally, Chapter 7 summarizes the contributions and future work of this research.
Chapter 2

Background

In this chapter, we review background information necessary to understand the research performed in this dissertation. We also review some of the prominent work related to ours to provide context of how our work fits into current research in distributed sensing and reasoning. This area of research is commonly referred to as distributed interpretation [38] and falls into two main categories: low level sensor fusion tasks and high level situational awareness. Our proposed solution can be used for low level sensor fusion as well as higher level information fusion or a combination of the two.

2.1 Agents and Multiagent systems

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators [53, 51]. The agent’s behavior is described by an agent function that takes as input the percepts from the agent’s sensors and outputs the actions the agent takes in the environment via its actuators (2.1).

![Conceptual agent model as presented in [53].](image)

Figure 2.1: Conceptual agent model as presented in [53].
This abstraction is useful since it describes living organisms such as humans, as well as computational agents that encompass virtual (software) and robotic (hardware) agents.

*Intelligent agents* are agents that act rationally by choosing actions that are expected to achieve its goals or maximize its utility. The specific features that define an intelligent agent are: (1) it has an internal model that represents its prior knowledge and state of the world it is designed for; (2) it has a set of goals or preferences of the state of the world; (3) it senses the environment it operates in and updates its internal model to be consistent with the observations; and (4) it takes actions based on its belief in the state of the world and that are expected to achieve its goals. We say the agent reasons about the state of the world to produce desirable outcomes.

In this dissertation we only consider intelligent computational agents, and not natural agents such as people, animals, etc. From here onwards we will simply refer to intelligent computational agents as agents.

It follows from the description of an intelligent agent that the three main tasks of an agent are:

1. **Sensing**: Using its sensors to make observations
2. **Reasoning**: Reasoning about the true state of the environment
3. **Acting**: Take actions based on goals and belief of true state of environment

Our research focuses on the first two tasks, where an agent reasons about the state of the world based on evidence it receives either from local sensors or messages from other agents. Whether the agents utilize their internal models using goal based planning or the principle of maximum expected utility is outside the scope of this dissertation.

A *multiagent system* (MAS) is characterized as a system composed of more than one agent where the agents communicate directly or indirectly through interactions
in the environment.

Often the domain world of the agent is too large for one single agent to sense and act upon the state of the world to achieve its goals. For example, consider an intrusion detection system for a large corporate computer network system. A single agent monitoring the network would need to monitor each server’s logs, firewalls, routers, incoming and outgoing network connections, email systems, database accesses, web server traffic, etc. The agent would need to have details on various particular vendors of these systems so as to be general enough to be a commercial product. This is a very challenging task and centralizing these tasks would require high computational and communication cost as well as agent complexity. To manage the complexity, the tasks could be divided up amongst several agents rather than one large agent. The process is to modularize a system into separate functional units and assign a functional unit to an agent. The agents communicate with each other to share information in order to achieve the overall system goal.

In a multiagent system, agent communication is often necessary since observations that are useful may be sensed by an agent that has direct access to the required sensor, and other agents may require this information. The observing agent can communicate the observation to the interested agents. Observations that are directly sensed by an agent are called local observations (evidence) and observations that are from other agents are called external observation. The observations are often referred to as evidence.

When designing a multiagent system the following key considerations must be addressed [64] :

• How do agents represent knowledge?

• How do agents communicate?

• Whom should agents be allowed to communicate with?
• What is the purpose of the communication?

• How should messages be processed?

• Is global consistency desired, and if so how is global consistency maintained?

These questions will be addressed in this dissertation with the proposal of a multiagent system where the agents represent their knowledge using Bayesian networks and communicate via messages that represent shared beliefs. The details of this agent model will be discussed further in the following chapters.

Agents can be either cooperative or selfish. Cooperative agents do not attempt to deceive or gain advantage over other agents, rather they work with other agents to achieve mutual goals. Selfish agents, however, will act in their best interest at the expense of other agents. In this dissertation, we only consider cooperative agents that work together to reason over the state of the environment.

2.2 Uncertain Reasoning

In this section, we discuss the problem of uncertain reasoning. Often intelligent agents must deal with uncertainty. The source of the uncertainty may be inherent in the environment they operate in, due to incomplete information, or from noisy sensors. Representing a complete internal model of the world is rarely feasible, and approximate or incomplete models must be used. The agent must therefore act under uncertainty.

The uncertain knowledge is encoded as degrees of belief in propositions. Several solutions have been developed to reason about uncertainty, such as weighted rule-based approaches, Dempster-Shafer theory, and Possibility theory. Many authors have argued that the use of probability to represent uncertain knowledge is the only method that leads to rational decision making [11, 46, 14, 52]. In this dissertation, our intelligent agents use probability to represent their degrees of belief.
In probabilistic reasoning systems, the world is divided into a set of random variables that represent the distinct aspects of the world that are relevant. Each random variable has a domain of values that it can take on. An assignment of values to all the random variables is an atomic event that represents one state of the world. The set of all atomic events is mutually exclusive and exhaustive. Each atomic event is assigned a probability that represents the degree of belief the world is in this particular state, and summarizes the uncertainty due to ignorance. A joint probability distribution assigns probabilities to all the atomic events and must sum to one. From the joint probability distribution, various forms of reasoning and inferencing can be performed using probability calculus.

Utilizing a joint probability distribution for knowledge representation suffers from three main problems: 1) intractability of acquisition of the probability table; 2) intractability of inferencing over the joint probability distribution; and 3) it is often unnatural and difficult to specify the joint probabilities. Bayesian networks [46, 26, 43] were developed to address these challenges by decomposing the joint probability distribution as individually specified conditional probability tables that exploit conditional independence in the model. Specifying conditional probability tables is more natural and easier for knowledge engineers to compute. Several algorithms have been devised to perform efficient inferencing over the Bayesian network by avoiding computation of the full joint probability table. Bayesian networks are discussed further in the next section.

**Bayesian Networks**

A Bayesian network is an efficient knowledge representation that encodes a joint probability distribution over a domain of propositional variables, where the model exploits the conditional independence relations among the variables. Each variable
in the domain is a random variable that has a finite set of states as its outcome\(^1\). The dependence relationships among variables are represented as a directed acyclic graph (DAG), where the nodes represent variables and the edges denote a direct influence of one variable on another. The strength of the relationship is encoded as a conditional probability table (CPT), \(P(X|\pi(X))\), where \(X\) is a node in the DAG and \(\pi(X)\) are the nodes parents. The joint probability distribution can be recovered as the product of the CPTs due to the conditional independence assumptions that are implicit in the network. In this way, the Bayesian network is a decomposed representation of a joint probability distribution. A formal definition of a Bayesian network is now presented using the definition from Neapolitan [43].

**Definition 2.2.1 (Bayesian Network).** Let \(V\) be a finite set of propositional variables defined on the same probability space, let \(P\) be their joint probability distribution, and let \(G = (V, E)\) be a DAG. For each \(v \in V\), let \(\pi(v) \subseteq V\) be the set of all parents of \(v\) and \(d(v) \subseteq V\) be the set of all descendants of \(v\). Furthermore for \(v \in V\), let \(a(v) \subseteq V\) be \(V \setminus (d(v) \cup \{v\})\), that is the set of propositional variables in \(V\) excluding \(v\) and \(v\)'s descendants. Suppose for every subset \(W \subseteq a(v)\), \(W\) and \(v\) are conditionally independent given \(\pi(v)\); that is if \(P(\pi(v)) > 0\), then

\[
P(v|\pi(v)) = 0 \text{ or } P(W|\pi(v)) = 0 \text{ or } P(v|W \cup \pi(v)) = P(v|\pi(V)).
\]

Then \(BN = (V, E, P)\) is called a Bayesian network.

The Bayesian network as defined above, encodes a unique joint probability distribution and relies on the conditional independence assumption, the so called *Markov property* that requires each variable be conditionally independent of its non-descendants given its parents. The chain rule for Bayesian networks states how to recover the joint probability distribution.

\(^1\)In this dissertation we only consider random variables with discrete states.
**Theorem 2.2.1** (Chain rule for Bayesian networks [26]). Let $BN$ be a Bayesian network over propositional variables $V = \{v_1, ..., v_n\}$. Then $BN$ specifies a unique joint probability distribution $P(V)$ given by the product of all conditional probability tables specified in $BN$:

$$P(V) = \prod_{i=1}^{n} P(v_i|\pi(v_i)).$$

A common task in reasoning with Bayesian networks is the calculation of the posterior probability distribution over a set of query variables, given a set of observations on evidence variables. This task is commonly called probabilistic update, or probabilistic inference. First the joint probability distribution, $P(V)$, is updated to respect the hard evidence, $e$. This can be achieved by multiplying $P(V)$ by the evidence $e$ and normalizing, resulting in the updated distribution, $P(V|e)$. From the updated distribution, the query can be calculated using marginalization.

**Definition 2.2.2** (Evidence). Evidence is a collection of findings on variables in a Bayesian network. There are three types of finding on a variable: hard findings, soft finding, and likelihood findings. A hard finding is an observation of the possible states of a variable or that it is in a particular state, which can be represented as a potential of ones and zeros. A soft finding is a specification of a probability distribution over the states of a variable. A likelihood finding is the likelihood ratio of the states of a variable. Hard evidence is a collection of hard findings, whereas soft evidence is a collection of soft finding, and virtual evidence is a collection of likelihood findings.

This chapter will mainly be concerned with hard evidence. Soft evidence and virtual evidence require special handling and will be discussed further later in this section and in Chapter 5.

**Definition 2.2.3** (Normalization). A probability table is normalized if it is divided by the sum over all the variables. We say the sum is a normalization constant.
Definition 2.2.4 (Marginalization). Given a joint probability distribution \( P \) over a set of propositional variables \( V \), \( P(V) \), we can construct a distribution over a subset of variables, \( W \), by summing away the variables not in \( W \):

\[
P(W) = \sum_{V \setminus W} P(V)
\]

The naïve inferencing method described above requires space and time that are exponential in the number of variables in the network. For a network of only a handful of variables, inferencing becomes intractable. In the next section we describe algorithms for exact inferencing that avoid the calculation of the full joint probability distribution during probabilistic inference. These algorithms run in exponential time and space in the worst case, but can be efficient even for large networks.

The complexity of inferencing is related to the tree-width of the Bayesian network graph. In human designed Bayesian networks, the tree-width is typically low and inferencing can be tractable even for very large networks. It has been shown that the computational complexity of exact inferencing to calculate the belief a variable is in a particular state, \( P(X = x) \), is \#P-complete [13]. The decision version of exact inferencing to decide \( P(X = x) > 0 \) is NP-complete [9].

The following example illustrates modeling using a Bayesian network.

Example 2.2.1 (Cold or Angina?). I wake up in the morning with a sore throat. I may either have a cold or I may suffer from angina. Angina is inflammation in the throat and can be mild or severe. Both a cold and angina can cause a sore throat and fever, while angina usually causes yellow spots to form in the throat.

To design a Bayesian network it is recommended to create the links as representing direct casual relationships, where parent nodes are causes and child nodes are effects. Such a Bayesian network is referred to as a casual Bayesian network. This method often results in more intuitive models that have smaller conditional probability tables that are easier to compute [25].
In our example, we first identify the six propositional variables: Sore Throat, Cold, Angina, Fever and Spots. Each of the variables, \{Sore Throat, Cold, Fever, Spots\} have states \{yes, no\} that represent the truth of the proposition. The variable Angina has states \{no, mild, severe\} to represent the severity or absence of angina.

Identifying the causal relationships among the variables, we find that a Cold can cause Sore Throat and Fever, while Angina can cause Sore Throat, Fever and Spots. The resulting Bayesian network is shown in Figure 2.2. The conditional probability tables that need to be specified are: \(\text{P}(\text{Cold})\), \(\text{P}(\text{Angina})\), \(\text{P}(\text{Sore Throat}|\text{Cold, Angina})\), \(\text{P}(\text{Fever}|\text{Cold, Angina})\) and \(\text{P}(\text{Spots}|\text{Angina})\). Before specifying the conditional probability tables, the conditional independence relations of the network should be checked to make sure they are correct. We will assume they are for this example, but if they were insufficient, other edges or nodes could be introduced as necessary. For example, if Spots were found to be a cause of Sore Throat, an appropriately directed edge could be added to capture the influence. We will not specify the conditional probability tables for this example, but note that the probabilities can be calculated by statistical data, experience, or subjective belief.

![Figure 2.2: Cold or Angina? Bayesian network](image-url)
Probabilistic Inference

When an agent receives new observations on the state of variables in its knowledge base, it updates its beliefs to reflect the new evidence. A central task of an agent is to compute the posterior probabilities on propositions of interest, given evidence it has received about the state of world. The first methods discussed concern belief updating when the evidence received are hard findings on a set of variables.

Belief update in the presence of hard evidence is carried out by conditioning. Two well known algorithms are presented that avoid the calculation of the full joint probability distribution. The first is Variable Elimination [67, 15, 68] a query or direct inference algorithm. The second is Lazy propagation [39] an all-marginals inference algorithm. The variant we will present is based on the popular Junction Tree algorithm, but we note it can function on any computational structure that preserves the original d-separation properties of the Bayesian network. Lazy propagation is a hybrid exact probabilistic update algorithm that merges ideas from query based and all-marginal methods.

As discussed previously, inferencing over a joint probability distribution requires an exponential amount of space and time. When the desire of an agent is to calculate the posterior probability distribution for only a set of query variables, the well known variable elimination algorithm can be used to avoid the exponential cost of inferencing. This algorithm is now described.

Variable Elimination

The variable elimination algorithm is used to calculate the posterior probability of a set of query variables given a set of evidence. The algorithm reduces this task by exploiting the axioms of marginalization [54]. By taking advantage of independence induced by the evidence scenario, as well as barren variables, the number of calculations can be reduced further by eliminating factors that have no influence on
the calculation of the query. To determine evidence induced independence, a graph based algorithm can be used to determine the \textit{d-separation} properties in the Bayesian network. If two variables are d-separated, then they have no influence on each other. The efficiency of variable elimination is dependent on the order the variables are marginalized. This is known as the \textit{elimination order}.

\textbf{Definition 2.2.5 (d-separation).} Variables $X$ and $Y$ in a Bayesian network are d-separated if for every path connecting $X$ and $Y$ there is an intermediate variable $Z$ such that one of the following is true:

- The connection is serial (Figure 2.3(a)) or diverging (Figure 2.3(c)) and there is evidence on $Z$.
- The connection is converging (Figure 2.3(b)), and neither $Z$ nor any of $Z$'s descendants have received evidence.

If two variables are not d-separated, then they are d-connected.

![Figure 2.3: d-separation connection types.](image)

To test whether a set of nodes $Q$ is d-separated from a set of nodes $R$ given observation variables $O$ can be determined by analyzing the paths of a moralized ancestral graph constructed from: $Q \cup R \cup O$. If all paths connecting $Q$ and $R$ intersect $O$, then $Q$ and $R$ are d-separated given $O$.

\textbf{Definition 2.2.6 (Barren Variable).} A variable in a Bayesian network is a barren variable if it is not a query variable, does not have evidence and all of its descendants are barren variables.
Barren variables, have no influence on the calculation of the posterior probability distribution of the query variables. This follows from the unity-potential axiom:

$$\sum_H P(H|T) = 1$$

We say $H$ are the head variables or conditioned variables and $T$ are the tail variables or conditioning variables.

We present a dynamic programming variable elimination algorithm that takes a set of potentials as input and eliminates all variables not in the requested query. The output is the set of potentials relevant to the query. If this algorithm is used to calculate the posterior marginal of a query $Q$, given evidence $O$ for a Bayesian network representation of a joint probability distribution, then the marginal is retrieved as the product of the returned set of potentials and normalized to retrieve $P(Q|O)$. This algorithm will be used to calculate messages in lazy propagation described in the Lazy Propagation section of Section 2.2.

**Definition 2.2.7 (Potential).** A potential, $\phi$, is a nonnegative function over its domain variables, $dom(\phi)$. Conditional and joint probability distributions can be represented as potentials, e.g. $\phi(X,Y) = P(X|Y)$.

**Algorithm 2.2.1 (Variable Elimination).** Let $BN = (V,E,P)$ be a Bayesian network, $\Phi$ be a set of potentials over $V$, $Q \in V$ a set of query variables, $O \in V$ be a set of evidence variables, and $E$ an elimination ordering for the variables $V - \{Q \cup O\}$.

1. Set $\Phi_R = \{\phi \in \Phi | \exists v \in dom(\phi) \text{ where } v \text{ is } d\text{-connected to } Q \text{ given } O\}$

2. Remove from $\Phi_R$ all potentials containing only barren head variables to obtain $\Phi_R'$.

3. For each $e \in E$:
   
   a) Marginalize $e$ out of $\Phi_R'$:

---

RAW TEXT END
1. Set $\Phi_e = \{ \phi \in \Phi_R | e \in \text{dom}(\phi) \}$

2. Let $\phi_e^* = \sum_e \prod_{\phi \in \Phi_e} \phi$

3. Update $\Phi_R' = \{ \phi_e^* \} \cup \Phi_R' - \Phi_e$

4. Return $\Phi_R'$

Junction Trees

Pearl pioneered an efficient message passing algorithm for calculating all single variable marginals in a Bayesian network [46]. The approach was shown to only work when the Bayesian network is a tree, but was later improved by Neapolitan to work in polytrees (singly connected graphs) [43]. Lauritzen and Spiegelhalter [36] proposed a message passing scheme that functions for any Bayesian network. This method constructs a secondary tree of cliques structure from a directed acyclic graph, where this new structure admits correct message passing similar to Pearl’s method. The method was further improved by Jensen et al. [23] where the secondary structure is called a Junction Tree.

Definition 2.2.8 (Domain Graph). A domain graph $G = (V,E)$ for a set of potentials $\Phi$ is an undirected graph where the nodes of $G$ are the variables in $\Phi$, $V = \bigcup_{\phi \in \Phi} \text{dom}(\phi)$. The edges in $G$ connect each pair of variables that are both members of the domain of a potential in $\Phi$, $E = \{(X,Y) | X, Y \in \text{dom}(\phi), \phi \in \Phi\}$.

Definition 2.2.9 (Junction Tree). A junction tree, $T = (C,E)$, is a tree where the nodes $C$ are maximal cliques of a triangulated domain graph, $G$, created from a set of potentials $\Phi$. $T$ has the running intersection property: for any pair of nodes $V,W$ in $C$, all nodes on the path between $V$ and $W$ contain the intersection $V \cap W$. Each potential $\phi$ in $\Phi$ is uniquely assigned to a clique containing $\text{dom}(\phi)$. Additionally, each edge in $E$ has a separator attached that has the intersection of the edge nodes as
its domain. Each separator contains two mailboxes: a collect and distribute mailbox, that are used for message passing operations.

A Bayesian network can be transformed into a junction tree by first moralizing the Bayesian network DAG and then triangulating. The maximal cliques of this new graph form the nodes of the junction tree.

The junction tree maintains the independence relations of the original Bayesian network and has the property that if the junction tree is locally consistent then it is globally consistent. Consistency is achieved by propagating evidence throughout the junction tree and will be explain in the context of lazy propagation in Section 2.2.

**Definition 2.2.10** (Moral Graph). A moral graph is an undirected graph constructed from a directed acyclic graph where an undirected edge is added between nodes that share a child in the DAG and the direction of all the edges in the DAG are dropped.

The moral graph of a Bayesian network is equivalent to the domain graph over its potentials (CPTs).

**Definition 2.2.11** (Triangulation). An undirected graph is triangulated if all cycles of length \( > 3 \) have a chord.

If the graph is not triangulated, Algorithm 2.2.2 can be invoked to triangulate it by introducing *fill-in edges* as necessary.

**Algorithm 2.2.2** (Triangulation). *Repeatedly eliminate a simplicial node \(^2\) that has not yet been eliminated in domain graph \( G \). If no such simplicial node exists, find a candidate node that has not been eliminated and add fill-in edges between uneliminated neighbors to make it simplicial, then eliminate it.*

**Definition 2.2.12** (Triangulation Weight). The triangulation weight of a junction tree, \( T = (C, E) \) created from Bayesian network \( BN = (V, E, P) \) is the sum of the

\(^2\)A simplicial node is a node with a complete neighborhood.
size of each clique in $C$:

$$T_W = \sum_{c \in C} |c|$$

where the size of each clique, $|c|$, is defined as the product of the number of states of each variable in $c$.

The overall complexity of performing belief update using a junction tree is dependent on the triangulation weight. However, the problem of optimal triangulation with the smallest triangulation weight is known to be NP-complete [1]. Therefore, heuristics are used to find a near optimal triangulation. These include heuristics for choosing a candidate node to eliminate when no simplicial node can be eliminated. Proposed heuristics for choosing candidate nodes include minimum fill-in weight, minimum fill-in size, minimum clique weight, and minimum clique size. Kjærulff performed extensive empirical tests of various heuristics for triangulation [30]. Computing the triangulated graph with minimum fill-in edges was found to be NP-complete in [66].

**Algorithm 2.2.3** (Create Junction Tree). Let $BN = (G, E, P)$ be a Bayesian network.

1. Create moral graph $G'$ from $G$.

2. Triangulate the moral graph $G'$.

3. Find the set of maximal cliques $C$ in $G'$.

4. Create a spanning tree $T$ from $C$ that has the running intersection property.

5. Uniquely assign the conditional probability tables to the cliques in $T$ such that the domain of the CPT is a subset of the domain of the clique.

The spanning tree in step 4 of Algorithm 2.2.3 can be created by first creating a cluster graph, where each node is a clique in $C$ and each pair of nodes that has a non-empty intersection has a link labeled with the intersection. The spanning
tree $T$ can be created using a \textit{Maximal Spanning Tree} algorithm such as Prim’s or Kruskal’s \cite{10, 26}.

After the construction of the junction tree, the nodes have a set of potentials (the CPTs of $BN$) associated with them. In the original definition of junction trees in the HUGIN propagation scheme \cite{23}, the product of the potentials would be associated with each node, rather than the decomposed set of potentials. Lazy propagation takes advantage of this decomposition as will be described further in the next section.

Figure 2.4(a) shows a triangulated moral graph and Figure 2.4(b) the resulting junction tree for Example 2.2.1. The chosen sets of potentials associated with each clique are: $C1 = \{P(Cold), P(Fever\mid Cold, Angina)\}$, $C2 = \{P(Angina), P(SoreThroat\mid Cold, Angina)\}$ and $C3 = \{P(Spots\mid Angina)\}$.  

![Triangulated Moral Graph](#) ![Junction Tree](#)

Figure 2.4: The triangulated moral graph (a) and junction tree (b) for the Cold or Angina? example.

### Lazy Propagation

Lazy propagation \cite{39} is an efficient junction tree algorithm that utilizes d-separation properties of the original Bayesian network by maintaining a multiplicative decomposition of clique and separator potentials. It merges the ideas of belief update algorithms that compute all marginals and direct query-based algorithms that compute a marginal for a query and exploit evidence-induced d-separation.
Lazy propagation can be used with any computational tree structure that maintains the d-separation properties of the original Bayesian network. In this dissertation we will assume a junction tree structure, rather than the original Lauritzen-Spiegelhalter or the Shafer-Shenoy structures. In particular, two mailboxes per separator are used, one for messages sent during the collect evidence phase and the other for the distribute evidence phase [37]. Cliques and separators in the junction tree maintain sets of potentials and combination of potentials is delayed as long as possible to take advantage of d-separation and unity potential properties of the Bayesian network.

Lazy propagation consists of two phases: collecting evidence to the designated root clique, and distributing evidence from the designated root clique to the rest of the junction tree. Evidence is collected and distributed by message passing, where each message is a collection of potentials that are calculated using variable elimination.

Following is a description of the lazy propagation algorithm:

1. Construct a junction tree for the Bayesian network.
2. Apply hard evidence.
3. Designate a root node for junction tree.
4. Invoke Collect Evidence on designated root of junction tree.
5. Invoke Distribute Evidence on designated root of junction tree.
6. Invoke Calculate Posterior Marginals.

Hard evidence is incorporated by applying hard evidence on a variable $X$ with all cliques $C_i$ where $X \in \text{dom}(C_i)$. This is done to fully exploit d-separation properties of the Bayesian network induced by the evidence. Hard evidence on a variable $X = x$ is incorporated by the reduction of the domain of all potentials $\phi_i$ where $X \in \text{dom}(\phi_i)$.
to only include configurations of the potential where $X = x$. All configurations where $X \neq x$ are simply removed. This process is called an instantiation of $X$.

**Algorithm 2.2.4** (Collect Evidence). Let $C_i$ and $C_j$ be adjacent cliques in the junction tree and let $S$ be the separator between $C_i$ and $C_j$. Let $\Phi_{C_i}$ and $\Phi_{C_j}$ be the set of potentials associated with $C_i$ and $C_j$. Let $\Phi^\uparrow$ and $\Phi^\downarrow$ be the set of potentials stored in the collect and distribute mailboxes of $S$ respectively. If Collect Evidence is invoked on $C_j$ from $C_i$, then:

1. $C_j$ invokes Collect Evidence on all adjacent cliques except $C_i$.

2. The message $\Phi^\uparrow$ from $C_j$ to $C_i$ is calculated (using variable elimination on $\Phi_{C_j}$ with query $S$) and stored in the collect mailbox of $S$.

3. Update $\Phi_{C_i} = \Phi_{C_i} \cup \Phi^\uparrow$.

**Algorithm 2.2.5** (Distribute Evidence). Let $C_i$ and $C_j$ be adjacent cliques in the junction tree and let $S$ be the separator between $C_i$ and $C_j$. Let $\Phi_{C_i}$ and $\Phi_{C_j}$ be the set of potentials associated with $C_i$ and $C_j$. Let $\Phi^\uparrow$ and $\Phi^\downarrow$ be the set of potentials stored in the collect and distribute mailboxes of $S$ respectively. If Distribute Evidence is invoked on $C_j$ from $C_i$, then:

1. The message $\Phi^\downarrow$ from $C_i$ to $C_j$ is calculated (using variable elimination on $\Phi_{C_i}$ with query $S$) and stored in the distribute mailbox of $S$.

2. Update $\Phi_{C_j} = \Phi_{C_j} \cup \Phi^\downarrow - \Phi^\uparrow$.

3. $C_j$ invokes Distribute Evidence on all adjacent cliques except $C_i$.

**Algorithm 2.2.6** (Calculate Posterior Marginals). Let $P(X|\epsilon)$ be the posterior marginal of $X$. Calculation of marginals is then performed on each variable $X$ by the following:

1. For each variable $X$
a) Let $\Phi_X = \{\arg\min_{i} \Phi_{C_i} \mid X \in \text{dom}(C_i)\}$.

b) Invoke Variable Elimination on $\Phi_X$ with query $X$ to retrieve the relevant potentials $\Phi_{R_X}$.

c) Calculate:

$$P(X|\epsilon) = \frac{\prod_{\phi \in \Phi_{R_X}} \phi}{\sum_X \prod_{\phi \in \Phi_{R_X}} \phi}$$

Algorithm 2.2.6 can be modified to calculate posterior marginals using the separators as well as the cliques.

In the lazy propagation algorithm, marginals of all variables in the Bayesian network are calculated by first instantiating all variables with hard evidence, then performing the collect evidence and distribute evidence phases, known collectively as a full propagation of evidence. Finally, the single variable marginals are calculated using variable elimination on the appropriate cliques.

**Belief Revision**

Belief revision is the process of revising a joint probability distribution to be consistent with uncertain evidence. There are two general issues that must be addressed in belief revision methods:

1. How is the evidence specified?

2. How are the beliefs revised?

We review three main belief revision methods that are relevant to our work: Pearl’s virtual evidence method, Jeffrey’s rule, and iterative proportional fitting procedure. Pan et al. showed that when dealing with a single evidential finding, the virtual evidence method and Jeffrey’s rule can be viewed as IPFP with a single constraint [45]. As such, all three methods commit to a belief revision process that minimizes the belief change of the original distribution.
**Virtual Evidence Method**

The virtual evidence method [46] is used to revise beliefs when observations are provided as likelihood ratios of the states of a variable $X$. The likelihood ratios represent an observer's strength of confidence in the observation.

In practice, the virtual evidence can be incorporated into a Bayesian network by creating a dummy node, $VE$, that has the same states as $X$ with only $X$ as a parent. The conditional probability table $P(VE = v|X)$ is specified such that,

$$P(VE = v|X = x_1) : P(VE = v|X = x_2) : \cdots : P(VE = v|X = x_n)$$

conforms to the likelihood ratios. Hard evidence $VE = v$ is entered and propagation is performed. The virtual evidence method can be interpreted as: the probability of observing variable $X$ is in state $x_j$ given the true state is $x_i$. The uncertain evidence is converted into certain evidence on a virtual event, hence the name given by Pearl. The beliefs are revised through Bayes conditioning.

Virtual evidence method can be characterized as a “nothing else considered” [49] approach because the evidence specifies only the strength of the evidence before the evidence is accepted.

**Jeffrey’s Rule**

Jeffrey’s rule$^3$ [22] is a mechanism for updating a probability distribution given soft evidence. The rule is commonly stated as:

$$Q(A) = \sum_i P(A|B_i)Q(B_i)$$

where $Q(A)$ is the revised belief in $A$ and $Q(B)$ is soft evidence on $B$. The rule is valid only if $P(A|B)$ is invariant given the evidence, i.e. $P(A|B)$ is the same before

---

$^3$Jeffrey’s rule is sometimes referred to as the rule of *probability kinematics*. 

23
and after the evidence on $B$ is received. Jeffrey’s rule can be characterized as an “all things considered” [49] method because the evidence specifies the effect on beliefs after the evidence is accepted.

Several authors [46, 47, 8, 45, 50] have observed that virtual evidence method can be viewed as formally equivalent to the likelihood ratio version of Jeffrey’s rule. This result admits the conversion of soft evidence into virtual evidence and vice versa.

**Iterative Proportional Fitting Procedure**

The Iterative Proportional Fitting Procedure (IPFP), originally proposed in [16], is an iterative method of revising a probability distribution to respect a set of given probability constraints in the form of soft findings (i.e. posterior marginal probability distributions over a subset of variables). Convergence of IPFP was proven in [12] and [58] for the discrete case.

Let $P(V)$ be the original probability distribution defined over a set of domain variables $V$ and $P(S_1), ..., P(S_k)$ be soft evidence on $\{S_1, ..., S_k\} \subseteq V$. Then one iterative step of IPFP is defined as:

$$Q_0(V) = P(V)$$

$$Q_i(V) = Q_{i-1}(V) \cdot \frac{P(S_j)}{Q_{i-1}(S_j)}$$

where $j = (i - 1) \text{ mod } k + 1$.

The procedure performs cycles of $k$ steps, one for each constraint, until convergence is reached$^4$. IPFP minimizes the $I$-divergence of the original distribution with the revised distribution that respects all the constraints.

$^4$If the probability constraints are incompatible with the distribution, then convergence is impossible and the procedure will oscillate. See [58] for some methods of dealing with inconsistent evidence.
\textit{I-divergence} (also known as Kullback-Leibler distance or cross-entropy) is a measure of the distance between two joint probability distributions $P(V)$ and $Q(V)$:

\[ I(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \]

We say $Q^*(V)$ is an $I_1$-projection of $P(V)$ on a set $Q = \{P(S_1),...,P(S_k)\}$ of probability constraints if $Q^*(V)$ minimizes $I(P||Q)$ amongst all distributions that satisfy $Q$. Therefore, IPFP converges to the $I_1$-projection of $P(V)$ on $Q$.

Chan and Darwiche proposed a new distance measure \textit{CD-distance} [7, 13] for measuring belief change. They note that even though \textit{I}-divergence is a popular measurement of the distance between two joint probability distributions, it is not a true distance measure since it is not symmetric. Further, \textit{I}-divergence only captures an average-case bound on belief changes.

The CD-distance measure is a true distance measure\footnote{A true distance measure must satisfy the three properties of distance: positivity, symmetry, and the triangle inequality.} that captures a worst-case bound on belief change, which Chan and Darwiche argue provides a more accurate measure of distance and is easier to calculate. CD-distance is calculated as:

\[ CD(P, Q) = \log \max_x \frac{Q(x)}{P(x)} - \log \min_x \frac{Q(x)}{P(x)} \]

where $0 \frac{0}{0} = 1$ and $\frac{\infty}{\infty} = 1$.

In this dissertation, we will compare beliefs in our simulations using both \textit{I}-divergence and CD-distance.
2.3 Related Work

Distributed Bayesian Networks

Multi-Entity Bayesian Network (MEBN) [35, 33], proposed by Laskey et al. extends the expressiveness of Bayesian networks through the instantiation of parameterized Bayesian network fragments (MFrags) that are composed into a situation-specific Bayesian network (SSBN), using a knowledge-based network construction mechanism. The MEBN framework allows for a distributed representation of a Bayesian network, but after the SSBN is constructed, reasoning is centralized using standard Bayesian inference. Therefore, MEBN is not appropriate for multiagent distributed interpretation tasks.

Semantically-Linked Bayesian Networks (SLBN) [44] is a framework for performing probabilistic inference over individually designed Bayesian networks. SLBN defines linkages between semantically similar variables where probabilistic influences are propagated over variable linkages from one Bayesian network to another using soft evidence and virtual evidence. Soft evidential update is performed using the wrapper methods of [45].

SLBN is proposed as a general distributed inferencing method, and does not address use in a multiagent environment. Further, SLBN does not address the problem when linkages result in multiply connected Bayesian networks. Therefore, the rumor problem is not solved or addressed.

The advantage of SLBN over our proposed work is the lack of need for a common ontology. However, SLBN requires a mapping function to related similar concepts amongst the linked Bayesian networks. Our work assumes this mapping has been implicitly performed during the design of each AEBNs internal model.

In the next chapter, we discuss distributed Bayesian network models under a multiagent paradigm, and propose our model.
Sensor Networks

The Kalman filter [28, 53, 41] is a common technique used in the fields of sensor and data fusion to estimate the state of a linear dynamic system corrupted by white Gaussian noise using noisy sensor observations over time. The method maintains two multivariate Gaussian distributions that represent the prediction estimate of the current system state given the previous knowledge of the state and a measurement estimate of the current state given the current observations, which are fused together at discrete time steps to form a superior estimate of system state. The fused state estimate is calculated using a weighted average of the two estimates that weights more heavily the estimate with the smallest expected error.

The Kalman filter respects a Markov assumption - where the current state is dependent on only a finite history of previous states. The Kalman filter is therefore analogous to a hidden Markov model (see Figure 2.5), where the main difference is in a hidden Markov model, the probability distributions are discrete, whereas they are continuous in a Kalman filter. Further, the Kalman filter’s expressiveness is greatly restricted when compared to hidden Markov models, or general Bayesian networks and only relations that are Gaussian in nature can be represented. Several extensions to the Kalman filter have been proposed, such as the relaxation of the white Gaussian noise assumption or the accommodation of non-linear systems.

![Figure 2.5: The Kalman filter represented as a hidden Markov model for a linear dynamic system with internal state $X_t$, and sensor observation $Z_t$.](image)

Figure 2.5: The Kalman filter represented as a hidden Markov model for a linear dynamic system with internal state $X_t$, and sensor observation $Z_t$. 

27
One particular extension of the Kalman filter that is related to our work is the Covariance Intersection algorithm [27]. This extension allows for a decentralized data fusion system, where multiple estimates of a common quantity from distributed processors are fused together without any prior knowledge of the cross-correlation between the estimates. The processors are arranged into an undirected network structure and communicate mean and covariance estimates to their neighboring processors. Each processor first calculates its local partial estimate using local sensor information using a standard Kalman filter and then propagates this estimate to neighboring processors. After receiving messages from its neighbors, each processor fuses its local estimate with the received estimates using a convex combination of their mean and covariances.

The convex combination method has the property that the resulting covariance always lies within the convex hull of the points of intersection of the covariances of the original estimates, resulting in a superior estimate irrespective of the correlation amongst the individual estimates. Effectively, redundant information has no effect on the fusion process and does not corrupt the system. Therefore, redundant information can be ignored because of the properties of convex combination of Gaussian distributions, thus allowing for flexible network topologies to be considered.

In general sensor networks, Utete’s work investigates decision making in decentralized sensing networks, where each node is a sensor that is capable of processing data, communicating and making decisions [55]. Nodes are organized into a peer-to-peer network, where no node has central control or global knowledge of the network topology, and nodes only communicate with their neighbors. Each node fuses information received from neighboring nodes to retrieve the global information of the entire network. Correct function of the network requires no sensor information is redundantly incorporated during fusion, thereby introducing bias.

In decentralized sensing networks, bias can be corrected for or avoided only if
the network is fully connected or a tree. Constraining the network to a tree or fully connected graph is a very restrictive limitation. The scalability of a fully connected network is severely limited, while a tree structured network is vulnerable to failures that disconnect subnetworks. Utete and Durrant-Whyte proposed a network management algorithm [56] that allows multiply connected network topologies for network robustness, but restricts communication only over “active” links that result in a spanning tree topology. If a network link fails, network management will activate new links to route around the failure and maintain correct operation of the system.

We claim that our proposed work allows for more flexible network topologies using a communication solution to avoid bias (rumors).

**Truth Maintenance Systems**

A justification-based truth maintenance system (JTMS) is a form of nonmonotonic belief revision system where the beliefs are restricted to propositional logic representations. In a JTMS, inferred assertions are dependent on the direct assertions used to assert it during inferencing. Each inferred assertion maintains a “justification set” of assertions that justify the inferred assertions truth. When new facts are introduced or retracted, logical consistency is maintained by introducing or retracting assertions. Each assertion that has a relationship to the new facts, update their justification set by adding or removing assertions to be consistent with the new facts. If the justification set becomes empty then the proposition is labeled OUT to represent that it no longer holds in the JTMS knowledge base. If they do hold, they are labeled IN.

JTMS is similar to hard and soft evidential update in Bayesian networks, in that the belief system is revised to respect the new evidence and be consistent with it. This is a non-monotonic belief update, where previously held beliefs may be reduced or increased based on the new evidence. In the case of JTMS, the beliefs must be 0 or 1, while a Bayesian network is more general and allows degrees of beliefs in [0,1].
Since each assertion is dependent on a set of justifying assertions in the JTMS, we can represent the dependence using a DAG, where parent nodes represent direct causal relationships to the children nodes. This is analogous to the dependence structure of causal Bayesian networks and means we could represent a JTMS as a Bayesian network where the conditional probability tables represent the disjunction of the supporting propositions in the JTMS. Each node represents a proposition and has two states: IN and OUT to represent whether the proposition holds in the knowledge base. To perform belief update, new facts can be entered as hard evidence in the Bayesian network and standard belief propagation can be performed to revise the knowledge base.

Belief update in Bayesian networks with virtual evidence is a different method of incorporating uncertain evidence. It does not hold the same properties as hard and soft evidence where the evidence is treated as a constraint on final beliefs. Virtual evidence only specifies the strength a piece of evidence has on a belief system and therefore is not treated as a constraint on the final beliefs. Only the likelihood ratio of the virtual evidence is respected. In this way belief update with virtual evidence does not correspond to JTMS.

Multiagent Truth Maintenance (MTM) proposed by Huhns and Brigeland [21] is a cooperative multiagent framework where each agent reasons non-monotonically using justification-based truth maintenance systems (JTMS) to maintain the integrity of its own knowledge base. Agents communicate by exchanging messages that are assertions over shared variables. A distributed truth maintenance algorithm determines whether global inconsistencies amongst agents need to be resolved and provides a mechanism for resolving them. The cost of ensuring complete global consistency of all the agents’ knowledge bases makes this property impractical. Instead MTM aims for a weaker form of global consistency by ensuring consistency of shared data amongst agents.

Our extended AEBN multiagent system shares similar belief revision goals as
MTM, such as maintaining global consistency of shared beliefs amongst the agents. In both systems, agents partition their internal models into private and shared variables. The shared beliefs are exchanged between agents via message passing, while the private variables are internal to an agent. Agents receiving messages revise their internal models to be consistent with the received beliefs. The belief revision process in both models commit to a principle of minimizing the change of belief necessary to ensure consistency.

MTM differs from AEBNs in that a receiving agent can reject received information if it conflicts with strongly held local beliefs. In AEBN, we assume publishing agents have oracular or expert knowledge over their shared variables and subscribing agents must respect the publishers beliefs. However, it is permitted for an AEBN to discount a publishing agent’s belief by explicitly modeling the agent’s reliability directly into its internal knowledge model. Additionally, the publisher/subscriber nature of the AEBN system restricts agent communication to a DAG topology, whereas in MTM communication can be bidirectional or contain cycles.

Finally, in an MTM multiagent system reasoning is based on propositional logic and therefore the system does not support reasoning about uncertainty and is not concerned with the rumor problem. Our multiagent model is based on uncertain reasoning using Bayesian networks and therefore must address the rumor problem. While there are similarities in the belief revision methods of the two multiagent systems, they are essentially aimed at addressing completely different problem domains.
Chapter 3

Probabilistic Multiagent Systems

This chapter is concerned with multiagent systems that represent their knowledge using probability and share their beliefs with other agents in the system. We review a prominent approach to probabilistic knowledge representation in multiagent systems, and then introduce our agent model.

As Pan identifies, the three main issues of probabilistic multiagent systems are [44]:

1. How is a joint probability distribution decomposed amongst the agents?

2. How are beliefs or local observations exchanged amongst agents?

3. How is global consistency maintained in the system?

The assumptions and constraints defined for a probabilistic multiagent model to address these three main issues lead to different formalisms with various advantages and disadvantages. In this dissertation, we will highlight some of the differences between our agent model and others proposed.

3.1 Multiply Sectioned Bayesian Networks

Multiply Sectioned Bayesian Networks (MSBNs) [64, 61] is a knowledge representation formalism for multiagent uncertain reasoning that effectively sections a large Bayesian network into subnetworks that are each assigned to an agent in the multiagent system.
The MSBN knowledge representation formalism is built up from five guiding basic assumptions of an “ideal” probabilistic multiagent system. These assumptions are used to derive the requirements and constraints necessary that give rise to MSBNs. For discussion on how the assumptions lead to the MSBN formalism see [64, Chapter 6]. The five basic assumptions of MSBNs are:

1. Each agent’s belief is represented by probability.

2. Agent’s communicate with concise messages that are joint probability distributions over the variables they share.

3. A simpler agent organization is preferred in which agent communication by concise message passing is achievable.

4. Each agent represents its knowledge dependence structure as a DAG.

5. Within each agent’s subdomain, a JPD is consistent with agent’s belief. For shared variables, a JPD supplements an agent’s knowledge with others’.

Xiang argues the logical consequence of the basic assumptions is a hypertree structure that is built from a multiply sectioned directed acyclic graph (hypertree MSDAG). Each node in the hypertree represents an agent and the DAG represents the Bayesian network that models the agent’s subdomain of knowledge. Thus as opposed to having one distinct, locally designed, Bayesian network that is encapsulated within each agent, each agent effectively contains a piece of a larger, globally designed, Bayesian network. Agents communicate only with neighbors in the hypertree by passing messages that are made up of the variables shared among the agents. The agent interfaces d-separate the subdomains.

A formal set of definitions for MSBNs are presented below using the definitions from Xiang [64].
**Definition 3.1.1** (Hypertree MSDAG). Let $G = (V, E)$ be a connected DAG sectioned into subgraphs $\{G_i = (V_i, E_i)\}$. Let the $G_i$’s be organized as a connected tree $\Psi$, where each node is labelled as $G_i$ and each link between $G_k$ and $G_m$ is labeled by the 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 $G_i$ and $G_j$ in $\Psi$ (the running intersection property). Then $\Psi$ is a hypertree over $G$. Each link between the subgraphs of $\Psi$ is a hyperlink. A hypertree MSDAG is a hypertree if each node $x$ contained in more than one subgraph, there exists a subgraph $G_i$ that contains its parents, $\pi(x)$.

**Definition 3.1.2** (MSBN). An MSBN $M$ is a triplet $(V, G, P) : V = \bigcup_i V_i$ is the total universe where each $V_i$ is a set of variables called a subdomain. $G = \bigcup_i G_i$ (a hypertree MSDAG) is the structure where nodes of each subgraph $G_i$ are labeled by elements of $V_i$. Let $x$ be a variable and $\pi(x)$ be all parents of $x$ in $G$. For each $x$, exactly one of its occurrences (in a $G_i$ containing $\{x\} \cup \pi(x)$) is assigned $P(x | \pi(x))$, and each occurrence in other subgraphs is assigned a uniform potential. $P = \prod_i P_i$ is the JPD, where each $P_i$ is the product of the potentials associated with nodes in $G_i$. Each triplet $S_i = (V_i, G_i, P_i)$ is called a subnet of $M$. Two subnets $S_i$ and $S_j$ are said to be adjacent if $G_i$ and $G_j$ are adjacent in the hypertree.

Distributed inferencing in a MSBN multiagent system is performed by compiling the hypertree MSDAG into a linked junction forest, which is a tree of junction trees. An inferencing scheme analogous to message passing in junction trees is used to collect and distribute evidence amongst agents, ensuring global consistency in the system.

MSBN was originally proposed for distributed inferencing over a global Bayesian network by assigning subnetworks to individual processors for efficient inferencing on computationally constrained equipment [60]. Xiang argues MSBNs are also appropriate for multiagent systems where the subnetworks can be individually designed as long as soundness of sectioning is achievable.
The MSBN framework is appropriate if the multiagent system can be compiled into a linked junction forest. This is a highly restrictive constraint on the multiagent organization and the internal knowledge model of each agent. To compile the agent system into a linked junction forest, the hypertree MSDAG first must perform a distributed moralization and triangulation procedure over all the agents to ensure consistency of the agent’s graphical models. The triangulation has a partial order constraint in order to ensure a linkage tree can be constructed, which is a data structure used for efficiency of communication. Once this step is complete, each agent constructs a local junction tree for efficient inferencing over its subdomain. From the local junction trees and the linkage trees, the original hypertree MSDAG is converted into a linked junction forest over which system level inferencing is performed.

MSBNs have the requirement that the union of the local DAGs of agents must also be a DAG. In order to ensure this, a distributed verification process must be performed to ensure the acyclicity of the union of each agent’s DAG. If directed cycles exist, the composition of the agents is invalid.

When variables exist in more than one subnet, only one subnet that contains the complete family specifies the conditional distribution of the variable. This is necessary to ensure local and global consistency in the agent system. The method proposed is to assign the “correct” conditional probability table to one agent, and a uniform distribution to all others. Determining the CPT to respect requires intervention from a system designer, or a complex negotiation scheme. However, in individually designed agents, there may not exist an agent that contains the complete family of an interface variable. Xiang devised a distributed algorithm to determine if each interface variable meets this requirement. If this is not the case, the agent decomposition is not valid and agents may need to be merged, or their subdomains modified to satisfy this requirement.

The verification and compilation algorithms demonstrate the complexity involved
with probabilistic reasoning in multiagent systems. The complexity is required if one commits to the five basic assumptions as Xiang shows admirably. However, the restrictions imposed by MSBN semantics restrict the autonomy of the agents, and imposes a tight coupling of the agents that limits their applicability in distributed reasoning. Xiang has made several arguments to support the role of MSBNs for multiagent systems [65, 62, 61, 64, 63], however, individually designing the agents can be a challenging task (and may not be possible) in order to satisfy the requirements of the MSBN model. Xiang provides little guidance on how to design agents to satisfy the restrictions, rather only methods for checking whether the design is sound.

MSBN is focused on the distributed computation of a global Bayesian network, where each agent is responsible for a sub-network. In our proposed system we designate agents with expert knowledge over particular variables and the sharing of this knowledge among interested agents. The internal models of each agent are the concern of each agent’s designer. This stresses the autonomy, rather than the distributed computational aspects of the multiagent system. We only require consistency over the shared variables to support normative system behavior. Their hidden (non-shared) variables represent an agent’s internal beliefs about the world and therefore are not relevant outside the agent.

Our approach is distinct from Xiang’s in that we commit to a different set of basic assumptions, resulting in a different formalism. Our model is less complex as it does not require the agents to be organized into a hypertree MSDAG, but instead introduces a strong independence relation called the oracular assumption. Xiang describes MSBNs as a tightly coupled framework [65], it is our goal to set out to construct a loosely coupled framework that stresses the autonomy of the agents. Additionally, Xiang shows that concise message passing is only achievable in tree topologies. We adopt Xiang’s basic assumptions, but we relax basic assumptions #2 and #3 in order to allow agents to be organized in more complex topologies than trees.
Further, we introduce a new basic assumption, the oracular assumption, that ensures global consistency is achieved via message passing in multiply connected graphs.

MSBNs are more expressive than our agent model and are more appropriate for agent systems that must coordinate. They also are based on conditional independence relations in the system and do not require additional independence assumptions not present in Bayesian networks, whereas in our model we introduce an oracular assumption. Additionally, since the agents conform to a tree topology, MSBN does not have a rumor problem. Our agent model allows for more flexible topologies than trees, hence we must detect and compensate for rumors to avoid bias. These topics and their implications will be discussed in the rest of the dissertation.

The focus of this work is the design of a multiagent system where the agents are loosely coupled resulting in multiagent systems that are simpler to design and implement. Our proposed framework is more flexible in agent topology than MSBNs, where in MSBNs agents are restricted to a hypertree structure that has the running intersection property. In our framework, agents can be organized as multiply connected graphs, with some restrictions on communication as will be discussed in the next section.

3.2 AGENT ENCAPSULATED BAYESIAN NETWORKS

We now present our agent model, which extends Bloemeke’s Agent Encapsulated Bayesian Network model. First, we briefly provide an overview of Bloemeke’s work and restrictions we aim to address. Finally, we present a formal description of our extended model.

Original Formalism

This research extends the Agent Encapsulated Bayesian Network (AEBN) multiagent model originally proposed by Bloemeke [3]. This model describes a method of linking
individually designed Bayesian networks using a multiagent framework, where each agent utilizes a Bayesian network to represent its internal knowledge representation of the world. The agents communicate by passing messages that are represented as probability distributions over shared variables.

The agents in this model are organized in a publisher-subscriber hierarchy, where the topology of agent communication must conform to a DAG structure. Agents pass messages from publisher to subscriber that are single marginal probability distributions over variables they share. Producer agents are assumed to be experts or more knowledgeable about the variables they produce and share this information with subscribing agents via message passing. Subscribing agents integrate the beliefs of the publishing agents by revising their internal model so it is consistent with the publishing agent. This is done simply by replacing the agent’s current view of the shared variable with the publishers. In this way, the publishing agent is said to have oracular knowledge over the variables they produce.

The originally proposed method of revising an agent’s Bayesian network relied on the assumption of independence of all received evidence, \( E \). The belief revision was performed using Jeffrey’s rule to update \( P(V) \) with \( Q(E) \):

\[
P(V) = \frac{P(V) \prod_{E_i \in E} Q(E_i)}{P(E)}
\]

This restriction as well as the limitation of only passing single marginals can result in a loss of dependence among shared variables in two possible ways. We illustrate these two cases with the following two examples. In the first example, consider an agent \( X \) sends its beliefs of variables \( A \) and \( B \) to an agent \( Y \), shown in Figure 3.1(a). Since only single marginals are passed, agent \( Y \) will receive \( P(A) \) and \( P(B) \) losing any dependence between them.

In the second example, agent \( X \) and \( Y \) send their beliefs in variables \( A \) and \( B \) respectively to agent \( Z \), shown in Figure 3.1(b). The revision procedure will result in
forcing $A$ and $B$ to be independent in agents $Z$’s internal model after absorbing the passed messages. Should we not commit to a revision procedure that minimizes the change in agent $Z$’s model but respects the beliefs of agent $X$ and $Y$?

Valtorta et al. [57] argue that agents should not force independence of the received evidence, and illustrate their argument with the following example:

**Example 3.2.1.** Consider an agent that models the age group $(A)$ and education level $(E)$ of US citizens. The agent conducts a small survey on a sample of the population and calculates a joint probability distribution, $P(A,E)$. Later, the agent communicates with two other agents that provide it with accurate US census data for age groups, $Q(A)$, and education levels, $Q(E)$. If we treat the evidence as independent in the receiving agent, then the agent revises its model $P(A,E)$ as $Q(A,E) = Q(A) \cdot Q(E)$. In doing so, it loses all information from its survey!

Instead, Valtorta et al. argue the agent should instead revise its model to a joint probability distribution, $Q^*(A,E)$ such that:

- The marginals for the US census agents are respected.

- The distribution $Q^*(A,E)$ is the distribution that is closest to the original distribution $P(A,E)$ (measured as $I$-divergence).

---

1The example was inspired by Demming and Stephans 1940 paper on IPFP [16].
Soft evidential update can be used to satisfy these requirements by revising an agent’s joint probability distribution to respect the beliefs received from publishing agents.

The independence restrictions limit the expressiveness of AEBNs. We propose an extension of AEBNs to address these problems in the next section.

Message passing in multiply connected agent graphs results in the well known rumor problem. Bloemeke proposed a method of identifying redundant influences in an AEBN network, and two methods of compensating for the rumors. The first is a communication solution that extends message passing by passing joint probability distributions. The second is a model-based solution. No formal proofs of correctness of the solutions were provided. Further, the solutions were devised with the assumption of independence of received evidence, and are not appropriate for our agent model. We propose a communication solution that is effective in the context of our agent system and prove its correctness under an assumption of coherence.

Proposed Framework

In our proposed agent model, each agent represents its internal knowledge base as a Bayesian network. Each agent’s probabilistic model is partitioned into three sets of variables:

- **Input variables (I):** variables which other agents have better knowledge
- **Output variables (O):** variables which this agent has the best knowledge and that are shared with other agents
- **Local or hidden variables (L):** variables which are private to this agent

**Definition 3.2.1** (Agent Encapsulated Bayesian Network). An AEBN is defined as a tuple: \( A = (I, L, O, E, P) \), where \( I \) is a set of input variables that other agents have
better knowledge of, $L$ is a set of local variables and $O$ is a set of output variables that the agent has oracular knowledge of. The union $V = I \cup L \cup O$ define the variables of the AEBNs local Bayesian network, where $E$ is the edges in the model that define the causal relationships amongst the variables $V$ and $P$ is the unique joint probability distribution defined over $V$. The union $S = I \cup O$ are the AEBNs shared variables, and $L$ are its private (non-shared) variables.

Our desire is to ensure global consistency of shared variables, while minimizing the changes to each agent’s local model. This is achieved by treating messages as soft evidence and utilizing soft evidential update.

Each agent provides its best guess as to the correct distribution of its input variables, and relies on other more knowledgable agents providing it with a more accurate view. In the event, no agent can be found, or communication is severed, the overall system will gracefully degrade due to the agents using their estimated guesses or last received belief from the knowledgeable agent.

Agents communicate via passing of messages that are joint probability distributions over their shared output variables, $O$. The topology of the communication in the multiagent system must conform to a DAG structure to ensure equilibrium can be reached. The agents are organized into a publisher/subscriber hierarchy, where agents are publishers of their output variables and subscribers to their input variables. The underlying assumption is known as the oracular assumption, where one agent is more knowledgable about certain variables and shares its knowledge with interested agents. It is permissible for multiple agents to share knowledge over the same quantities, however, each quantity must have its own unique label.

The method of updating an agent’s probability distribution upon the receipt of messages from other agents is described in a related paper [57], where the messages are called soft evidence. In particular, we adopt the modeling approach of introducing observation variables into an agents Bayesian network, and updating the agent’s
probability distribution using the approach of soft evidential update \cite{57, 6}. Therefore, each agent that receives messages from other agents obtains soft evidence for one or more observation variables\textsuperscript{2} (see Figure 3.2) that are created by the following procedure:

1. Create an observation variable, $\text{Obs}_i$, for each soft evidence received, where the states of the observation variable correspond to the possible outcomes of the soft evidence.

2. Add directed edges to $\text{Obs}_i$ from all variables in the Bayesian network that have a direct influence on the observation. The set of parents $d$-separates the observation variable from the rest of the network.

3. Model the logical dependence of the parents of $\text{Obs}_i$, $\pi(\text{Obs}_i)$, on $\text{Obs}_i$ by specifying the conditional probability table $P(\text{Obs}_i|\pi(\text{Obs}_i))$ where

$$P(\text{Obs}_i = o|\pi(\text{Obs}_i) = \vec{x}) = 1 \iff \vec{x} \text{ corresponds to } o.$$ 

Figure 3.2: Introduction of an observation variable in a subscriber agent for absorption of a publisher message over shared variables $I_1, \ldots, I_k$.

To update an agent’s distribution $P(V)$ with new evidence $Q(E_1, E_2, \ldots, E_n)$ for some set of observation variables $\{E_1, E_2, \ldots, E_n\} = I$ one calculates the joint probability $P(V)$, dividing by the marginal probability $P(I)$, and multiplying it by the

\textsuperscript{2}The introduction of observation variables is a modeling technique that enables update on a single observation node, rather than a set of nodes.
new distribution of \{E_1, E_2, ..., E_n\}, this corresponds to the application of Jeffrey’s rule,

\[ Q(I) = Q(E_1) \cdot Q(E_2) \cdot ... \cdot Q(E_n), \]  

(3.2.1)

thus obtaining:

\[ Q(V) = P(V \mid I|I) \cdot Q(I) = \frac{P(V)}{P(I)} \cdot Q(I). \]  

(3.2.2)

In the case in which the input variables are not independent in the receiving agent, Equation 3.2.1 does not hold. (See [57, Section 5] for a detailed discussion on this point.) Lemma 1 in [57] allows the replacement of Equation 3.2.2 by:

\[ Q^*(V) = P(V \mid I|I) \cdot Q(I) = \frac{P(V)}{P(I)} \cdot Q_I^*(I), \]  

(3.2.3)

where \(Q_I^*\) is the \(I_1\)-projection of probability distribution \(P\) on the set of all distributions defined on \(I\) and having \(Q(E_i), i = 1, ..., n\), as their marginals. In practice, \(P(V)\) could be updated to \(Q^*(V)\) using the *big clique algorithm* of [57, 29], *lazy big clique algorithm* of [32], or the *wrapper methods* of [45].

Thus a mechanism similar to that already used for updating probabilities in a Bayesian network adjusts the world view of the agent, \(P(V)\), into a conditional probability table \(P(O|I)\). Note that this table is calculated using the local observations of the agent: \(P(O|I) = \sum_L P(O, I, L)/P(I)\). It then combines that table with the external view of the inputs, \(Q(I)\), to allow the calculation of the new values for the output variables \(Q(O)\).

Given this view of the purpose of each agent in the overall system, an agent system may be considered an expansion of the Bayesian network formalism to a DAG where the distribution of the variables of one agent is obtained by conditioning on its input variables. This is not strictly the case for two reasons. First, when input variables are not independent in the receiving agent, then the calibration equation 3.2.2 must
be replaced by the formally identical, but substantially and computationally more complex equation 3.2.3.

Second, the oracular assumption imposes the additional constraint that, in the agent system, unlike a Bayesian network, all parents are not affected by their descendants. More precisely, the only variables that may affect the variables in an agent are (1) those in the agent itself and (2) those in a preceding agent. In order to provide a formal definition of “preceding agent,” we introduce the notion of communication graph in Section 3.2.

**Communication Graphs**

In order to represent the message passing and updating implications of AEBN’s, we define a graphical representation of the agent system, called a *communication graph*. This graph is a DAG whose nodes are the agents and where edges are drawn from a publisher of shared variables to each of the subscribers of the shared variables. These edges are in turn labeled with the variables that they share. It is permissible for an agent to subscribe to only a subset of the published variables of another agent. In this case, the publishing agent will marginalize $Q(O)$ to the desired subset and pass this marginal to the subscriber agent.

We can now formalize the constraint that, in the agent system, all variables that are parents are not affected by their descendants. Let $A_i$ and $A_j$ be two distinct agents, let $V_i$, $V_j$ be the sets of variables in agent $A_i$ and $A_j$, respectively, and let $W_i \subseteq V_i$, $W_j \subseteq V_j$. Then if there is no directed path in the communication graph from $A_j$ to $A_i$, any changes (whether by observation or by intervention) in the state of the variables in $W_j$ does not affect the state of the variables in $W_i$. This is a very strong condition on the distribution of the variables in different agents of the agent system. This is *not* a symmetric relation, and therefore cannot be represented by any independence relation, since every independence relation is symmetric. There
is an analogy to be made with casual Bayesian networks [48]. In a causal Bayesian network, when a variable is set (by external intervention), the parents of that variable are disconnected from it; more precisely, the result of the intervention is to create a new Bayesian network in which we remove the edges incoming into a variable that is set. The analogy, however, is not complete. In a causal Bayesian network, when a variable is set by intervention, some of the parent variables may be affected through backdoor paths, as explained in [48, section 3.3]. In an AEBN, there is no possibility for a variable in an agent to be affected by a descendant agent.

Consider as an example a four-agent system, where a supervisor agent fuses reports from two observer agents, each of which reports information from a single sensor agent. The communication graph shown in Figure 3.3 is constructed by first identifying shared variables ($S$, $L_1$, and $L_2$), then directing labeled edges from the producing agents to the consuming agents. The labels for the edges correspond to the shared variables. In this example, the edges directed from the Sensor agent to the Observer\textsubscript{1} and Observer\textsubscript{2} agents are labeled with $S$, and the edges from Observer\textsubscript{1} and Observer\textsubscript{2} to the Supervisor agent are labeled with $L_1$ and $L_2$, respectively. Henceforth this example will be referred to as the Redundantly Observed Sensor Example (ROSE).

![Figure 3.3: Redundantly Observed Sensor Example (ROSE) communication graph.](image-url)
Chapter 4

The Rumor Problem

A central problem of message passing in probabilistic systems is the familiar rumor problem, where cycles in message passing cause redundant influence of beliefs. This problem is often known as the “rumor problem.” In this chapter, we develop algorithms to identify and solve the rumor problem in the context of our multiagent system. Our identification algorithm adopts the approach proposed by Bloemeke [3]. In particular, we adopt the notion of a redundancy graph to identify the flow of rumors in the agent communication graph. To compensate for rumors, a communication solution is proposed that expands message passing. The solution highlights the challenges of correct message passing in probabilistic systems when rumors are present.

4.1 Redundant Influences

In decentralized systems, detecting and solving the rumor problem is critical to support normative decision making that is free of bias. Utete provided a useful “dining table” analogy (originally proposed by F. Banda) to highlight the effect rumors have in decentralized systems [55]. Consider a dinner party where guests sit around a large oval table. The table is so large that guests can only communicate with their immediate neighbors. Guests can glean information from remote guests only through the communication of intermediaries. The host of the dinner party communicates a rumor to their neighbors, who in turn communicate the rumor to their neighbors. Eventually the information arrives from both sides of the table to the guest opposite
of the host at the table. The dinner guest receives the same piece of information from two (assumed) independent sources, thus arriving at a biased view. If the guest were aware that both communications originated from the same source, the information may be handled differently. Figure 4.1 depicts the described analogy.

![Figure 4.1: Example of a rumor initiated by the dinner party host, labeled “Start”, spreading around a dining room table. The rumor reaches the guest labeled “End” from two distinct neighbors, doubly influencing them.](image)

We now demonstrate the rumor problem in the context of our agent model using the ROSE communication graph as an example.

It is within the communication graph that the nature of the rumor problem can be clearly understood. Using the ROSE communication graph as an example, we can see the problem centers on the fact that the supervisor agent’s view of the world, held in its Bayesian network, is doubly influenced by the initial sensor reading. The supervisor computes its belief in $L$ as

$$P(L) = \sum_{I} P(L|I)P(I)$$  \hspace{1cm} (4.1.1)

which expands to

$$P(L) = \sum_{L_1, L_2} P(L|L_1, L_2)\phi_{Ob_1, \phi_{Ob_2}}$$  \hspace{1cm} (4.1.2)

where $\phi_{Ob_1}$ and $\phi_{Ob_2}$ are messages the supervisor agent receives from $Observer_1$ and $Observer_2$. Expanding $\phi_{Ob_1}$ yields
\[
\phi_{Ob_1} = \sum_S P(L_1|S)\phi_{Sensor} \tag{4.1.3}
\]

with \( \phi_{Ob_2} \) being calculated as

\[
\phi_{Ob_2} = \sum_S P(L_2|S)\phi_{Sensor} \tag{4.1.4}
\]

Finally, expansion of \( \phi_{Sensor} \) yields

\[
\phi_{Sensor} = P(S) \tag{4.1.5}
\]

Substitution of equations 4.1.3, 4.1.4, and 4.1.5 into equation 4.1.2 leaves us with the following equation

\[
P(L) = \sum_{L_1, L_2} P(L|L_1, L_2) \sum_S P(L_1|S)P(S) \sum_S P(L_2|S)P(S) \tag{4.1.6}
\]

Finally, pulling the sums out leaves the following equation for \( P(L) \)

\[
P(L) = \sum_{L_1, L_2, S} P(L|L_1, L_2)P(L_1|S)P(S)P(L_2|S)P(S) \tag{4.1.7}
\]

In equation 4.1.7, \( P(S) \) is redundantly incorporated in the supervisor agent, resulting in a redundantly influenced calculation of \( P(L) \), because the correct expression of \( P(L) \) (calculated using the chain rule) is

\[
P(L) = \sum_{L_1, L_2} P(L|L_1, L_2)P(L_1, L_2) \tag{4.1.8}
\]

Since there exists no directed path between \( L_1 \) and \( L_2 \) in the communication graph, they are independent by the oracular assumption and therefore:

\[
P(L_1, L_2) = \sum_S P(L_1|S)P(L_2|S)P(S) \tag{4.1.9}
\]
Substitution of equation 4.1.9 into equation 4.1.8 leaves the correct equation for calculating $P(L)$

$$P(L) = \sum_{L_1, L_2, S} P(L|L_1, L_2)P(L_1|S)P(L_2|S)P(S) \quad (4.1.10)$$

where equation 4.1.10 is the desirable outcome of message passing in the agent system and equation 4.1.7 is the actual outcome. Further, this problem can be made arbitrarily worse simply by adding additional paths between the sensor and the supervisor agents.

Redundant influences arise in a communication graph whenever the combination of messages received by an agent causes the belief in some shared variable to be over included.

A principal objective of this research is to allow for the handling of the rumor problem in an automated fashion. This will be achieved using algorithms that first identify redundant influences using the communication graph (Section 4.2) and then using a communication based solution (Section 4.3) to eliminate them.

4.2 Identifying Redundant Influences

This section describes a method for identifying redundant influences in a communication graph. $V_i$ redundantly influences $V_j$ if there exists multiple node-disjoint paths from $V_i$ to $V_j$. There is a redundant influence between nodes $V_i$ and $V_j$ in some communication graph $G$ if either $V_i$ redundantly influences $V_j$ or $V_j$ redundantly influences $V_i$.

Redundant influences are external to an agent and are dependent on the communication graph topology, therefore, it cannot be assumed they will be known in advance when the agent’s internal model is designed. Hence, the redundant influences are orthogonal to the dependencies that are encoded in the Bayesian network.
that is contained within an agent and it is appropriate to support the processing of redundant influences separately from the construction of individual agents.

**Theorem 4.2.1** (Redundant influences occur on node disjoint paths). [3] Given an AEBN communication graph \( G = (V, E) \) with nodes \( V_i, V_j \in V \), where \( V_i \) is an ancestor of \( V_j \), it is sufficient to identify all node-disjoint paths from \( V_i \) to \( V_j \) in order to see all routes of redundant influence from \( V_i \) to \( V_j \).

Following is a skeleton of a proof of Theorem 4.2.1. First we characterize thoroughly the routes through which redundant influences arrive at a node in the communication graph.

Assume that we have a communication graph \( G \), as defined above, such that between \( V_i \) and \( V_j \) there are only \( n \) node disjoint paths \( \{ V_i \to V_{i1} \to \ldots \to V_{ik_i} \to V_j, \ldots, V_i \to V_{n1} \to \ldots \to V_{nk_n} \to V_j \} \), where \( k_i, 1 \leq i \leq n \) is the length of path \( i \) minus the endpoints (4.2). Further, assume we have more than \( n \) redundant influences. Clearly, since redundant influences must occur along some series of paths, there must be some paths that are not node-disjoint between \( V_i \) and \( V_j \) causing the additional redundancies. Each of these additional paths must take on one of four forms (assume \( p, q \in [1, n] \)):

1. It starts at one node along node-disjoint path \( q \) and ends at a different node along node-disjoint path \( p \). (Figure 4.3(a))

2. It starts at one node along path \( q \) and ends at node \( V_j \). (Figure 4.3(b))

3. It starts at node \( V_i \) and ends at a node along node-disjoint path \( q \). (Figure 4.3(c))

4. It starts and ends along node-disjoint path \( q \). (Figure 4.3(d))

In all of these cases, the additional redundant influences are due to a subgraph that does not include both \( V_i \) and \( V_j \). The subgraph effectively amplifies the redundancy.
between $V_i$ and $V_j$, but the cause is redundant influences in the subgraph caused by multiple node disjoint paths between two nodes in the subgraph. If we were to compensate for these additional redundancies in the subgraph then we are left with the redundant influences from the $n$ node disjoint paths between $V_i$ and $V_j$. In other words, we can recursively remove redundant influences in subgraphs between $V_i$ and $V_j$ and be left with $n$ redundant influences corresponding to the $n$ node disjoint paths. The remaining $n$ redundancies could be compensated for similarly. This is a recursive argument and means we can identify all redundant influences by examining all pairwise node disjoint paths in the graph.

The Create Redundancy Graph algorithm, described below, can be used to detect and label node disjoint paths in a communication graph. Once this algorithm has been run, a new graph, known as the redundancy graph, is constructed. The redundancy graph has the same nodes and edges as the communication graph, but its edge labels are expanded if and only if there are redundant influences. This graph will be used, along with the original communication graph, to compensate for redundant influences in the communication solution described in Section 4.3.

![Diagram](image)

**Figure 4.2**: $n$ node disjoint paths between $V_i$ and $V_j$.

**Algorithm 4.2.1** (Create Redundancy Graph Algorithm). Let $G = (V, E)$ be a communication graph and $R$ be a copy of $G$ that will serve as the redundancy graph. The edge labels of $R$ will be expanded as described below.
1. Let $G'$ be a copy of $G$ for use in a maximum-flow problem [10].

2. Modify $G'$ by replacing each node $v$ that has multiple incoming edges with two nodes $v_1$, $v_2$. Replace each incoming edge to $v$, $<x,v>$, with a new edge $<x,v_1>$, and each outgoing edge from $v$, $<v,y>$, with a new edge $<v_2,y>$. Finally, create a directed edge $<v_1,v_2>^1$.

3. For each pair of nodes $v_s$ and $v_t$ in $G$ take each variable $s_i$ that is produced by $v_s$:

   a) Designate $v_s$ as the source for the flow problem and $v_t$ as the sink for the flow problem in $G'$.

---

1Without this step, the algorithm would find edge-disjoint paths instead of node-disjoint paths.
b) Set the maximum flow of each edge in $G'$ to 1.

c) Set the maximum flow to 0 for all outgoing edges in $G'$ from $v_s$ that have a label that does not contain variable $s_i$ in $G$.

d) Run the maximum-flow problem on $G'$.

e) If the flow into the sink is greater than 1, then redundant influences exist between the two nodes:

i. For each edge in the maximum-flow problem solution that has flow greater than zero (and therefore is on a node disjoint path), add the shared variable $s_i$ to the label of the corresponding edge in the redundancy graph $R$.

4. Return the redundancy graph $R$.

It is well known that the maximum-flow problem can be used to find all node disjoint paths between a pair of nodes $v_s$ and $v_t$ [31, Chap. 16]. Hence the above algorithm will expand the edge labels of all node disjoint paths that a shared variable’s influence travels for each combination of $v_s, v_t$. From Theorem 4.2.1, these node disjoint paths are also the redundant influences and therefore the algorithm will identify all redundant influences and expand the labels of the edges each redundant influence travels.

The time complexity of the algorithm to solve the maximum-flow problem described in [31] is $O(nm \log(\frac{n^2}{m}))$ where $n = |V|$ and $m = |E|$. Since a maximum-flow problem is executed for each pair of nodes, the total time for the Create Redundancy Graph algorithm is $O(n^3m \log(\frac{n^2}{m}))$. This time is within $O(n^5)$ because $m$ in the worst case is $n^2$. We note that since this is a distributed system, algorithms exist to find the maximum flow in $O(n^2 \log^3 n)$ time using $O(n^2(\log^3 n + \sqrt{m}))$ communication complexity [42] but requires each node having knowledge of the communication graph (which can be accumulated in no worse than $O(m)$ time).
It is important to note that this is the only step in an AEBN system that requires global knowledge of the network topology. After the edges have been labeled no further knowledge outside of the immediate neighborhood is necessary.

Considering the ROSE example of Figure 3.3, the Create Redundancy Graph algorithm produces the redundancy graph of Figure 4.4. In this graph, only two edges have expanded labels. This arises because only in the case of the cycle (i.e., $v_s = \text{Sensor}, v_t = \text{Supervisor}$) will the flow problem return a flow greater than 1. In this case, all four of the edges between the Sensor node and the Supervisor node will have a flow of 1 and therefore will have $S$ added to their edge label.

![Figure 4.4: The ROSE redundancy graph.](image)

### 4.3 Communication Solution

This section proposes a method of compensating for redundant influences where agent communication has been expanded to pass joint probabilities along the appropriately labeled links in the redundancy graph, without any change in the local Bayesian networks of each agent. Computing the expanded messages requires the probability update algorithm used by the agents to be flexible enough to allow the calculation of joint probabilities involving fixed input and output variables.

The calculation of joint probabilities is not trivial, especially in the presence of soft evidence. The soft evidential update algorithms such as the Big Clique [57] and Lazy Big Clique [32] are based on the junction tree method, and were designed to
compute all single variable marginals, but they can be used to compute one or several joint probabilities using techniques such as value or variable propagation, described in [24, Section 5.1]. Additionally, the wrapper methods [45] can be used similarly with a junction tree algorithm, or with a direct query-based algorithm such as bucket elimination.

Given the labeling in the redundancy graph, the correct (i.e., not redundantly influenced) probabilities can be retrieved at each agent without any modification in the Bayesian network it maintains, so long as the message that travels each edge is the joint probability of its label variables. Since no local Bayesian network model modification is necessary, this is known as the communication solution.

Care must be taken when removing redundant influences to ensure all redundant influences are correctly compensated for. This is done by ordering the removal of the redundant influences. Consider an example where an agent $a_i$ receives three messages from neighboring agents: $\phi_1(A, B, C)$, $\phi_2(A, B, D)$, and $\phi_3(A, E)$. Agent $a_i$ subscribes to the input variables $C$, $D$ and $E$, hence its local Bayesian network calculates $P(O|C, D, E)$ and therefore needs to calculate the joint probability, $Q(C, D, E)$, from the received messages. The three messages contain redundant influences: $\phi_1(A, B, C)$ and $\phi_2(A, B, D)$ have redundant influence $\{A, B\}$, while $\phi_3(A, E)$, $\phi_1(A, B, C)$, and $\phi_2(A, B, D)$ have redundant influence $\{A\}$ (4.5). If we first remove the redundant influence common to all, $\{A\}$, we can divide two of the messages by $Q(A)$ and have:

$$Q(B, C|A) = \frac{\phi_1(A, B, C)}{\sum_{B, C} \phi_1(A, B, C)} = \frac{Q(A, B, C)}{Q(A)}$$

$$Q(B, D|A) = \frac{\phi_2(A, B, D)}{\sum_{B, D} \phi_2(A, B, D)} = \frac{Q(A, B, D)}{Q(A)}$$

Next, we eliminate the remaining redundant influence, $\{B\}$, that is common to $Q(B, C|A)$ and $Q(B, D|A)$, by dividing one of the updated messages by $Q(B)$:
\[ Q(C|A, B) = \frac{Q(B, C|A)}{\sum_{A,C} \phi_1(A, B, C)} = \frac{Q(B, C|A)}{Q(B)} = \frac{Q(A, B, C)}{Q(A)Q(B)} \]

The correct joint probability is retrieved by combining the updated messages, \( Q(C|A, B), Q(B, D|A), \) and \( Q(A, E), \)

\[ Q(A, B, C, D, E) = Q(C|A, B)Q(B, D|A)Q(A, E) \]

Finally, the required joint probability of the inputs is calculated by marginalizing,

\[ Q(C, D, E) = \sum_{A,B} Q(A, B, C, D, E) \]

This is correct, if \( Q(A, B) = Q(A)Q(B), \) which means \( A \) and \( B \) are independent. However, this is not correct in general and therefore removal of the redundant influences in this order is incorrect. The removal of redundant influences must be ordered so as to not lose any dependence relations among the redundant influences. In this example, this can be achieved by first removing the redundant influence \( \{A, B\} \) from \( \phi_1(A, B, C), \) and finally removing the remaining redundant influence \( \{A\} \) from \( \phi_3(A, E). \) This can be restated, more generally, as ordering the received messages and removing the largest (in cardinality) remaining redundant influences. We now present formal definitions for this procedure, and then an efficient algorithm is described.

Figure 4.5: Multiple overlapping redundant influences.

**Definition 4.3.1.** Each message \( \phi_i \) received by an agent \( a_k \) in the communication graph is a joint probability distribution over the variables that make up the corresponding label in the redundancy graph. Let \( \text{dom}(\phi_i) \) be the domain of message \( \phi_i. \)
Definition 4.3.2. A pair of messages $\phi_i$ and $\phi_j$ received by agent $a_k$ contains potential redundant information if and only if $\text{dom}(\phi_i) \cap \text{dom}(\phi_j) \neq \emptyset$. We call $\text{dom}(\phi_i) \cap \text{dom}(\phi_j)$ the redundant influence or redundant information between $\phi_i$ and $\phi_j$.

Since each shared variable has a unique label, and agent messages are expanded to pass shared variables that cause redundant influences, if a pair of messages $\phi_i$ and $\phi_j$ contain redundant information, they will have one or more common shared variables, i.e. $\text{dom}(\phi_i) \cap \text{dom}(\phi_j) \neq \emptyset$.

Definition 4.3.3 (Redundancy-free update rule). A message $\phi_i$ is updated to remove redundant information by the following update procedure:

$$\phi_i^* = \frac{\phi_i}{\sum_{D_i \cap D_V} \phi_i}$$

where $D_i = \text{dom}(\phi_i)$, $D_V = \bigcup_{j \neq i} \text{dom}(\phi_j)$ and $\phi_j$ has not been updated. If $D_i \cap D_V = \emptyset$, then the update is defined as the identity of $\phi_i$.

The update rule states that removal of redundant information is achieved by dividing a message by the marginal of variables it has in common with other messages, excluding messages that have already been updated. It is necessary to exclude updated messages, because by definition they are no longer redundant with any other message. If the message does not share variables with any qualifying message, then the update does not change the message. Observe that if $D_i \cap D_V = \text{dom}(\phi_i)$, the resulting updated message is 1. Ex: $\frac{P(A)}{P(A)} = 1$. This can occur if the domain of a message is a subset of another message.

Definition 4.3.4. Let $\Phi$ be a set of messages received by agent $a_k$, ordered as $O = \phi_1, \phi_2, ..., \phi_n$, where $n$ is the number of messages in $\Phi$. We call the order $O$, the update order. If the redundancy-free update rule is applied to each $\phi \in \Phi$ according to the update order, then the resulting updated messages $\Phi^*$ are said to be redundancy-free.
We note that $\Phi^*$ does not depend on the update order, however, the order chosen will affect the efficiency of the update. We do not discuss update ordering considerations further in this research but note it is related to the variable elimination order problem.

In order to minimize the change of the receiving agents, independence relations in an agent’s Bayesian network should be respected. This means we should not treat the joint probability over the input variables as one big joint distribution, otherwise we may make some variables independent after update that were not independent before update.

An agent $a_k$ preprocesses its received messages by updating them to redundancy-free messages (Definition 4.3.4) which can be safely combined together to retrieve the uncontaminated joint over the input variables. This joint probability is marginalized to the required inputs which are joint distributions over dependent variables or single marginals (when no dependency exists). Since the calculation of a large joint distribution over all the inputs is often unnecessary and very expensive in both time and space, the following algorithm is a more efficient method that exploits the independence of the input variables and decomposes the calculation of the required inputs. The two methods result in the same calculated inputs:

**Algorithm 4.3.1** (Remove Redundant Influences). *Let $\Phi$ be a set of messages received by agent $a_k$.*

1. Partition $\Phi$ such that for each partition $s_i$, where $|s_i| > 1, \bigcap_{\phi \in s_i} \text{dom}(\phi) \neq \emptyset$.

2. For each partition $s_i$ where $|s_i| > 1$
   
   a) Order the messages in $s_i$

   b) Apply the redundancy-free update rule on the messages according to order
The messages can be processed in this manner due to Lemma 4.5.2 in Section 4.5, which proves independence of variables in messages that do not intersect. The result of the algorithm will be a set of messages that are redundancy-free. To absorb the messages into the agent’s Bayesian network, the messages in each partition need to be combined. Each combination is added as a piece of soft evidence, as formalized in the algorithm below.

**Algorithm 4.3.2 (Create Soft Evidence).** Let $S$ be the resulting partition of updated messages from the Remove Redundant Influences algorithm.

1. For each partition $s_i \in S$
   
   a) Combine the messages in $s_i$:
   
   $$\phi_i = \prod_{\phi \in s_i} \phi$$
   
   b) Marginalize to needed inputs:
   
   $$\phi_i = \sum_{\text{dom}(\phi_i) - I} \phi_i$$
   
   Each $\phi_i$ is treated as soft evidence, and absorbed by creating observation variables and using soft evidential update.

![Diagram](image)

**Figure 4.6:** The extended ROSE redundancy graph.

Returning to an extended version of the ROSE example, where an additional agent responsible for monitoring the environment communicates its belief in the current environmental conditions to the Supervisor agent (Figure 4.6). Consider the problem
of correcting the redundant influence on the Supervisor agent. The three incoming messages correspond to \( m(Observer_1) = P(L_1, S) \), \( m(Observer_2) = P(L_2, S) \) and \( m(Environ) = P(E) \) which lead to the double counting of \( P(S) \). To remove the redundant influence, the Supervisor agent invokes \textit{Remove Redundant Influences}, which first partitions the received messages, which in this example results in two partitions:

\[
s_1 = \{P(L_1, S), P(L_2, S)\}, s_2 = \{P(E)\}
\]

Since \(|s_1| > 1\), the messages are ordered as \( < P(L_1, S), P(L_2, S) > \) and the redundancy-free update rule is applied on the messages according to the order, resulting in the updated messages:

\[
s_1 = \{P(L_1|S), P(L_2, S)\}
\]

The Supervisor agent invokes \textit{Create Soft Evidence} to retrieve the appropriate soft evidence, which combines the updated messages in \( s_1 \) to give:

\[
P(L_1, L_2, S) = P(L_1|S)P(L_2, S)
\]

Which is marginalized to the required inputs:

\[
P(L_1, L_2) = \sum_S P(L_1, L_2, S)
\]

from which the desired probability \( P(L) \) can be computed as:

\[
P(L) = \sum_{L_1, L_2, E} P(L|L_1, L_2, E) \sum_S P(L_1|S)P(L_2|S)P(S)P(E)
\]

\[
= \sum_{L_1, L_2, E} P(L|L_1, L_2, E)P(L_1, L_2, E)
\]

where we note that \( P(L_1, L_2, E) \) is calculated either using equation 3.2.2 or 3.2.3.
4.4 Coherent AEBN Systems

The semantics of AEBN require the marginal belief of shared variables be the same in each agent due to the oracular assumption. Bias in the system is avoided by detecting and correcting for the propagation of rumors. However, in particular agent communication topologies, the oracular assumption is violated when rumors are compensated for. We will explore this problem with an example.

Consider the following AEBN system composed of five agents with a communication graph as shown in Figure 4.7(a). It is clear to see that information from agents $A$ and $B$ doubly influence agent $E$ through multiple node disjoint paths. To remove the rumors, the communication solution will expand the messages from agents $C$ and $D$ to pass joint distributions including $\alpha$ and $\beta$. The redundancy graph for this example is shown in Figure 4.7(b).

![Figure 4.7: The communication graph (a) and resulting redundancy graph (b) illustrating the coherency problem.](image)

In this example, agent $A$ shares its marginal belief, $P(\alpha)$, with agents $C$ and $D$. Likewise, agent $B$ shares its marginal belief, $P(\beta)$, with agents $C$ and $D$. Both agents $C$ and $D$ revise their internal models to be consistent with the beliefs received by treating the beliefs as soft evidence. Although $P(\alpha)$ and $P(\beta)$ will be respected in both agents, the joint probability $P(\alpha, \beta)$ is determined separately by each agent. In other words, the dependency between the variables $\alpha$ and $\beta$ are modeled separately in
agents $C$ and $D$. The only restriction on $P(\alpha, \beta)$ in each agent is that the marginals $P(\alpha)$ and $P(\beta)$ respect the evidence received.

According to the communication solution, to correct for the rumors in agent $E$, both agents $B$ and $C$ must pass a joint probability that includes $\alpha$, $\beta$. Agent $E$ compensates for the rumor by conditioning one of the received messages by the shared redundant information, $P(\alpha, \beta)$.

No matter which message agent $E$ conditions the oracular assumption will be violated. In either case, one of the marginals will not be the same as in the publishing agent.

Consider agent $E$ chooses to condition $P(\alpha, \beta, \gamma)$ received from agent $C$. The redundancy-free joint probability over $\alpha, \beta, \gamma, \delta$ is then calculated as:

$$P(\alpha, \beta, \gamma, \delta) = \frac{P(\alpha, \beta, \gamma)}{\sum_{\gamma} P(\alpha, \beta, \gamma)} P(\alpha, \beta, \delta) \quad (4.4.1)$$

The oracular assumption states that the belief $P(\alpha, \beta, \gamma)$ must be the same in subscribing agent $E$ as the publishing agent $C$. Since agent $E$ will revise its beliefs to be consistent with $P(\alpha, \beta, \gamma, \delta)$, its belief $P(\alpha, \beta, \gamma)$ can be calculated as:

$$P(\alpha, \beta, \gamma) = \sum_{\delta} P(\alpha, \beta, \gamma, \delta) \quad (4.4.2)$$

$$= \frac{P(\alpha, \beta, \gamma)}{\sum_{\gamma} P(\alpha, \beta, \gamma)} P(\alpha, \beta, \delta) \quad (4.4.3)$$

$$= \frac{P(\alpha, \beta, \gamma)}{\sum_{\gamma} P(\alpha, \beta, \gamma)} \sum_{\delta} P(\alpha, \beta, \delta) \quad (4.4.4)$$

However, there is no guarantee that $\sum_{\gamma} P(\alpha, \beta, \gamma) = \sum_{\delta} P(\alpha, \beta, \delta)$ since both are independently calculated in separate agents. The equality may not hold for several reasons:

1. Agents $C$ and $D$ have differing conditional probability tables.

2. Agents $C$ and $D$ have differing conditional dependence models.
3. Agents C and D have differing d-separation induced by local evidence received by each agent.

Therefore, agent E may not respect the evidence received from agent C, violating the oracular assumption. It is trivial to see that the oracular assumption is violated even if agent E conditions \( P(\alpha, \beta, \delta) \) instead of the first message.

Rumors can only be corrected for and global consistency maintained, if the AEBN system is coherent, i.e., no pair of agents have conflicting dependency relations amongst shared variables. We distinguish between two different types of coherence, strong coherence and weak coherence, in the following definitions.

**Definition 4.4.1 (Strong Coherence).** Two AEBN’s, \( A = (I_A, L_A, O_A, E_A, P_A) \), \( B = (I_B, L_B, O_B, E_B, P_B) \) are said to be locally coherent if the belief on their shared variables, \( S_A = I_A \cup O_A, S_B = I_B \cup O_B \) agree: \( P_A(S_A \cap S_B) = P_B(S_A \cap S_B) \). If a pair of agents are not locally coherent, they are said to be incoherent. An AEBN system is said to be strongly coherent if all pairwise AEBNs in the system are locally coherent.

**Definition 4.4.2 (Weak Coherence).** An AEBN system is said to be weakly coherent if the only locally incoherent agents are agents that share no common descendent.

In our AEBN model, we only desire the system be weakly coherent. It is permitted for agents to be locally incoherent if the conflicting information does not reach a common subscriber agent, causing a conflict. From this point forward we will refer to weak coherence as global coherence, unless the type of coherence needs to be distinguished.

We propose two different methods of ensuring global coherence in an AEBN system. The first method is a design time solution that imposes an additional constraint on an AEBN system. The first step of the design time solution is to identify potential sources of incoherence in the agent composition. The following algorithm analyzes
Figure 4.8: Example illustrating strongly and weakly coherent systems. If $A, B, C$ and $D$ are locally coherent then the system is strongly coherent. If only $C$ and $D$ are locally incoherent, then the system is weakly coherent.

the resulting redundancy graph of an AEBN system and marks sets of agents that are incoherent.

**Algorithm 4.4.1** (Detect Incoherence). Let $\mathcal{A}$ be a set of agents in an AEBN system, $C$ a communication graph for $\mathcal{A}$, and $R$ the resulting redundancy graph.

1. Let $O$ be a topological ordering of the agents $\mathcal{A}$ in $C$

2. Initialize $ANCESTORS(\mathcal{A}) = \emptyset$ for each $A \in O$

3. For each agent $A \in O$:
   
   a) Let $\Pi$ be the set of parents of $A$ in $C$

   b) Set $ANCESTORS(A) = \bigcup_{\pi \in \Pi} (ANCESTORS(\pi) \cup \{\pi\})$

4. For each agent $A \in R$:

   a) For each pair of incoming edges to agent $A$, where $I$, the intersection of the edge labels has cardinality $> 1$:

      i. Let $\{X,Y\}$ be the set of parents of $A$ that correspond to the pair of edges

      ii. Let $\Pi = ANCESTORS(X) \cap ANCESTORS(Y)$
iii. If $\Pi \neq \emptyset$ and $\exists \pi \in \Pi$ with an outgoing edge labeled $E$, where $I \subseteq E$, then $\{X,Y\}$ are locally coherent

iv. Else, mark agents in $\Pi$ that have no parents in $R$ with an outgoing edge labelled $E$, where $I \subseteq E$

5. If no agents have been marked, then the AEBN system is globally coherent

Algorithm 4.4.1 is used at design time to detect if the AEBN system is globally coherent, and if it is not, marks the nodes in the redundancy graph that are incoherent. The algorithm examines the redundancy graph and identifies multiple sources of a joint probability over shared variables. The worst case time complexity of this algorithm is $O(NE^2)$ where $N$ is the number of agents, and $E$ the number of edges in the communication graph. As discussed previously in this section, when multiple sources independently model dependency information over shared variables, the oracular assumption can be violated and prevent global consistency of the system. To prevent this from occurring, a new design constraint on AEBN systems is proposed, where only one agent is permitted to have oracular knowledge of the dependence relations amongst shared variables. This agent may not be the producer of the marginal beliefs of the shared variables, but is responsible for modeling the interactions amongst these shared variables, as is demonstrated in the running example in this section.

The coherence problem occurs when there exists two or more common ancestors between two or more agents. We formalize the coherence problem as follows:

Let $G = (V, E)$ be a DAG and let $C \in V$, $A = A_1, \ldots, A_{k-1}, A_k$, $B = B_1, \ldots, B_{m-1}, B_m$ where $k, m > 1$, $A \cap B \cap \{C\} = \emptyset$ and $A \cup B \in V$. Further, let each $B_i \in B$ be reachable from each $A_j \in A$ and $C$ be reachable from each $B_i \in B$. Figure 4.9(a) shows the graphical structure defined. We say the graph is coherent if there exists a directed path that includes all nodes in $A$ or a directed path that includes all nodes in $B$, otherwise the graph is incoherent.
As an example, Figure 4.9(b) shows a graphical structure where there exists a directed path in A, thus the graph is coherent.

Figure 4.9: Graphical structure that leads to incoherence is shown in figure a), if there exists a directed path that includes all the nodes in A or all the nodes in B then the graph is coherent as shown in figure b) and c).

From the oracular assumption, we know that each agent’s set of shared variables have unique labels but edge labels are expanded in the redundancy graph if multiple node disjoint paths exists between any two agents. Since an AEBN communication graph is a DAG, there exist no directed cycles. Therefore, if there exists two or more common ancestors between two or more agents, and the system is coherent, then there must exist a directed path between the common ancestors or between the agents (as shown in Figure 4.9(b) and 4.9(c)). This means one of the common ancestors must have an expanded label on an outgoing edge that contains the intersection. This is in fact what Algorithm 4.4.1 checks for and proves its sufficiency in determining coherence.

Running algorithm 4.4.1 on the redundancy graph in Figure 4.7(b), identifies that agents C and D are incoherent on $P(\alpha, \beta)$. To ensure global coherence of the system, we need to alter the agent composition so only one agent is responsible for determining $P(\alpha, \beta)$. Figure 4.10(a)-4.10(d) identify four possible design solutions that will result in a globally coherent system. In Figure 4.10(a), agents A and B are
merged into one agent that is responsible for \( P(\alpha, \beta) \) and shares this information with agents \( C \) and \( D \). Figure 4.10(b) designates agent \( C \) as being responsible for \( P(\alpha, \beta) \) and communication is altered so agent \( C \) passes this information directly to agent \( D \), making communication from agents \( A \) and \( B \) to \( D \) unnecessary. Figure 4.10(c) introduces a new agent in the system that is responsible for \( P(\alpha, \beta) \) and passing this information on to agents \( C \) and \( D \). Finally, in Figure 4.10(d) agents \( C \) and \( D \) are merged into a single agent which results in a tree topology for the communication graph and hence the rumor problem no longer exist. Re-running algorithm 4.4.1 on each of these design solutions\(^2\) returns they are globally coherent and while each has its own set of trade-offs, they all are sufficient to ensure global consistency of the system.

The second method proposed is a runtime solution. If restrictions of the problem domain prevent coherence being achieved at design time, then the AEBN communication graph can be modified to fuse the conflicting evidence. The fusion is performed by special “proxy” agents that are responsible for reconciling conflicting distributions at runtime. The reconciled distribution is then absorbed by the appropriate incoherent and subscribing agents. Figure 4.11 shows a proxy agent introduced to reconcile two incoherent agents. In this example, agents \( C \) and \( D \) are incoherent on \( P(\alpha, \beta) \). The proxy agent \( F \) is introduced into the communication graph and communication links are updated so agent \( C \) and \( D \) no longer communicate directly with agent \( E \), but rather, agent \( F \) acts as a proxy and communicates the redundancy-free belief of \( P(\gamma, \delta) \) to agent \( E \). This is necessary in order for agent \( F \) to first remove the incoherence by fusing the beliefs of agents \( C \) and \( D \). After agent \( F \) has fused the beliefs received, it communicates the fused belief \( P(\alpha, \beta) \) to agents \( C \) and \( D \). To maintain consistency throughout the network it is necessary for \( F \) to send messages

\(^2\)There are many different possible design choices that could be made to ensure global coherence. The chosen solutions are for illustrative purposes and not an exhaustive set.
Figure 4.10: Design solutions for example in Figure 4.7(a) that ensure global coherence.

back to agents C and D. This appears to violate the AEBN assumption that an agent’s belief cannot be affected by a subscribing agent. A proxy agent is a special type of agent that is permitted to violate this assumption in order to maintain global coherence in the system. After agents C and D receive the fused belief, they will revise their internal models to be consistent with this belief and recompute their messages retransmitting them to agent F. After this step, the agents C and D are locally coherent. Agent F can now safely remove any redundant influences between the received messages and calculate the message to send to agent E.

The following algorithm describes the process of adding necessary proxy agents to ensure runtime global coherence.
Algorithm 4.4.2 (Revise Communication Graph). Let $\mathcal{A}$ be a set of agents in an AEBN system, $C$ a communication graph for $\mathcal{A}$, and $R$ the resulting redundancy graph.

1. Let $I$ be the resulting incoherent agents determined by executing algorithm 4.4.1

2. For each set $I_i$ of locally incoherent agents in $I$:

   a) Instantiate a new proxy agent $P$ and add to $R$

   b) Associate a Fusion function with $P$

   c) Revise the communication links in $R$ to flow through proxy agent $P$:

      i. For each agent $A$ in $I_i$:

         A. Let $B = B_1, ..., B_{k-1}, B_k$ be the set of agents $A$ communicates with.

         B. Revise directed edges of the form $(A, B_i)$ in $R$ to $(A, P)$ if edge label contains incoherent information

         C. Create directed edges $(P, B_i)$ and add variables in label associated with $(A, B_i)$ to edge label. If a variable is already in edge label, then remove it from the label

Algorithm 4.4.2 instantiates a proxy agent and inserts it between sets of publishing agents that are locally incoherent and their subscribing agents. The communication edges are updated to route communication from the publishing agents to the subscribing agents through the proxy agent. Rumors are also removed by a proxy agent before sending messages on to subscribing agents, therefore it is only necessary for the communication edges between the proxy agent and subscribing agents to be labeled as the union of the edge labels associated with the publishing agents in the communication graph.
The task of the proxy agent is then to fuse incoherent messages received from the publishing agents and to remove rumors before passing the necessary information on to the subscribing agents on behalf of the publishing agents.

There are several different methods a proxy agent can use to fuse incoherent beliefs. The resulting belief can be calculated by the following different Fusion functions:

1. Calculating a weighted average of the distributions, where the weight represents a trust (or reliability) score of the publishing agents. The scores could be maintained through experience, or fixed by a designer. If all agents are trusted equally then the result is an average of the distributions

2. Designating one of the distributions as the true value, where the designation could be chosen randomly or through a trust mechanism, etc.

3. Modeling the fusion as a disagreeing experts problem [25, section 3.3.5].

After agents receive the fused beliefs, they revise their internal models to be consistent with the fused belief. This effectively removes any incoherence, resulting in a globally coherent system.

Figure 4.11: Runtime solution for example in Figure 4.7(a). Proxy agent $F$ is introduced to reconcile conflict between agents $C$ and $D$ over $P(\alpha, \beta)$. The grey arrows are communication links that are removed in the original communication graph, and dashed lines are new communication edges introduced.
4.5 Proof of Communication Solution

We now prove the correctness of the communication solution for removing redundant influences.

**Lemma 4.5.1.** The domain of a message $\phi$ is invariant when applying the redundancy-free update rule. The resulting updated message $\phi^*$ is a conditional probability table.

**Proof.** Each message $\phi$ is a joint probability table, $P(\text{dom}(\phi))$. If the $\text{dom}(\phi)$ does not intersect the domain of any other message that has not been updated, the update rule returns the identity, $\phi$, which is a conditional probability table with an empty set of conditionals: $P(\text{dom}(\phi)|\emptyset)$. Otherwise, by the definition of conditional probability,

$$P(X|Y) = \frac{P(X \cup Y)}{P(Y)}$$

which corresponds to the update rule, where $X \cup Y = \text{dom}(\phi), Y = \text{dom}(\phi) \cap \{\cup_j \text{dom}(\phi_j)\}$ where $\phi_j$ has not been updated. \qed

**Lemma 4.5.2.** Let $\Phi$ be a set of messages received by agent $a_k$. Let $V = \bigcup_{\phi \in \Phi} \text{dom}(\phi)$ be the variables of $\Phi$. By the oracular assumption the variables in $\text{dom}(\phi)$ of message $\phi \in \Phi$, are independent of variables not in $\text{dom}(\phi), V - \text{dom}(\phi)$.

**Proof.** We prove the lemma by contradiction:

Let $\phi_i, \phi_j \in \Phi$ be two messages received by agent $a_k$ respectively from agents $a_i$ and $a_j$ (4.12). Assume there is a variable $v_i$ only in $\text{dom}(\phi_i)$ and variable $v_j$ only in $\text{dom}(\phi_j)$, and $v_i$ is not independent of $v_j$.

From the oracular assumption, the beliefs of agent $a_i$ are unaffected by its children in the communication graph. Therefore, the only agents that may influence $a_i$ are its ancestors. From the stated assumption, $v_i$ is not independent of $v_j$. There must be a directed path from the agent producing $v_j$ to $a_i$.

There are two cases:
1. $v_j$ is produced by $a_j$ and $a_j$ is an ancestor of $a_i$ (Figure 4.13).

2. $v_j$ is produced by a common ancestor, $a_s$, of $a_i$ and $a_j$ (Figure 4.14).
In both cases, there exist multiple node disjoint paths between the agent producing $v_j$ and $a_k$. The redundancy graph will extend the labels of the edges of these paths with $v_j$. In the communication solution, agent message passing is extended to pass messages over the labels in the redundancy graph. Since $a_i$ is a node along one of these paths, the message $\phi_i$ sent to agent $a_k$ must contain $v_j$. This contradicts the assumption and proves the lemma.

\begin{theorem}
Let $\Phi$ be a set of messages received by agent $a_k$, and $\Phi$ has been updated to $\Phi^*$, a set of redundancy-free messages. Let $V = \bigcup_{\phi \in \Phi^*} \text{dom}(\phi)$. For all $v \in V$, there is only one message $\phi \in \Phi^*$ of the form: $P(X|Y)$, where $v \in X$, $X \subseteq V$, $Y \subseteq V - X$ and all other $\phi$ with $v \in \text{dom}(\phi)$ are of the form: $P(W|Z)$, where $v \in Z$, $Z \subseteq V$, $W \subseteq V - Z$. Less formally, this can be stated as: there exists only one message with $v$ as a conditioned upon variable, and in all other messages with $v$ in their domain, $v$ is a conditioning variable.
\end{theorem}

\begin{proof}
Since each $\phi \in \Phi^*$ is redundancy-free, one of the following holds:

1. $v$ only belongs to the domain of one message, $\phi$, and $\text{dom}(\phi)$ does not intersect with the domain of any other message. By Definition 4.3.3 and Lemma 4.5.1, the update of $\phi$ is the identity and therefore all variables in $\text{dom}(\phi)$ are on the left hand side of the conditioning bar. Since no other message shares variables with $\phi$ there exists only one message with $v$ in its domain and it is of the form $P(X|Y)$, where $X = \text{dom}(\phi)$, and $Y$ is the empty set.

2. $v$ only belongs to the domain of one message, $\phi$, and $\text{dom}(\phi)$ intersects the domain of other messages. By Lemma 4.5.1, $\phi$ has been updated as:

\[ \phi^* = P(X|Y) \]

where $X = \text{dom}(\phi) - Y$, $Y = \text{dom}(\phi) \cap \bigcup_{\phi_j \in \Phi^*, \phi_j \neq \phi} \text{dom}(\phi_j)$ and $\phi_j$ has not been updated. Since $v$ only belongs to $\text{dom}(\phi)$, $\phi$ cannot be conditioned on $v$.
and hence \( v \) is on the left hand side of the conditioning bar. Therefore there exists only one message with \( v \) in its domain and it is of the form \( P(X|Y) \), where \( v \in X \).

3. \( v \) belongs to the domain of more than one message. By Definition 4.3.4, each message \( \phi_i \in \Phi^* \) with \( v \in \text{dom}(\phi_i) \) will be updated in some order \(<\phi_{i_0}, ..., \phi_{i_k} >\) where \( k \) is the number of messages. Updating conditions each message on the intersection of its domain with all other messages that have not yet been updated. Since \( v \) belongs to the domain of all \( \phi_i \), all but the last message updated in the order will be conditioned on \( v \). The last message \( \phi_{i_k} \) either will intersect other messages that have not yet been updated and do not have \( v \) in their domain, or will have an empty intersection (and hence return identity). In either case, \( v \) will exist in only one message with the form \( P(X|Y) \) where \( v \in X \), and all other messages with \( v \) in their domain will have the form, \( P(W|Z) \), where \( v \in Z \).

From the three cases above, it follows that each \( v \in V \) is uniquely a conditioned upon variable in one message, and a conditioning variable in all other messages that contain \( v \) in their domain. \( \square \)

**Theorem 4.5.4.** If \( \Phi^* \) is a set of redundancy-free messages received by agent \( a_k \), then the joint probability \( P(V) \), where \( V = \bigcup_{\phi \in \Phi^*} \text{dom}(\phi) \) is:

\[
P(V) = \prod_{\phi \in \Phi^*} \phi
\]

**Proof.** We will show the combination of the messages in \( \Phi^* \) results in the joint probability over \( V \), \( P(V) \).

Order the messages in \( \Phi^* \) according to the reverse order redundancy-free update was invoked on the messages. We call this the *message order*. Let \( \phi_1, \phi_2, ..., \phi_n \) be the message order, where \( n \) is the number of messages in \( \Phi^* \).
According to the redundancy-free update rule, each message is conditioned on the variables it shares with other messages that are later in the update order. Since the messages have been ordered the reverse of the update order, we can rewrite the update as:

\[ \phi_i^* = \frac{\phi_i}{\sum_{\text{dom}(\phi_i) \cap \bigcup_{j=1}^{i-1} \text{dom}(\phi_j)} \phi_i} \]

Therefore, by Theorem 4.5.3, each message has a set of unique variables that are conditioned upon, \( R_i = \text{dom}(\phi_i) - \{\text{dom}(\phi_i) \cap \bigcup_{j=1}^{i-1} \text{dom}(\phi_j)\} \). We call \( R_i \) the residual set of an updated message. Let \( S_i = \text{dom}(\phi_i) \cap \bigcup_{j=1}^{i-1} \text{dom}(\phi_j) \), be the conditioning variables for message \( \phi_i \). From Lemma 4.5.1, the updated message is the conditional probability table:

\[ \phi_i^* = P(R_i|S_i) \]

Each \( R_i \) is a disjoint subset of \( V \), such that \( \bigcup_i R_i = V \). Order the residual sets according to the message order. We call this the residual sets order. Let \( R_1, R_2, ..., R_n \) be the resulting residual sets order, where \( R_1 \subseteq \text{dom}(\phi_1), R_2 \subseteq \text{dom}(\phi_2), ..., R_n \subseteq \text{dom}(\phi_n) \).

We need to show that,

\[ P(V) = P(R_n, R_{n-1}, ..., R_1) = \phi_n \phi_{n-1} ... \phi_1 \]

We prove this by using induction on the number of messages in \( \Phi^* \). For \( n=1 \) we have,

\[ P(V) = P(R_1) = \phi_1 \]

since there are no other messages intersecting \( \text{dom}(\phi_1) \), \( R_1 = \text{dom}(\phi_1) \) and the updated message \( \phi_1 = P(\text{dom}(\phi_1)) \).
We will assume the theorem holds for any residual sets order created from \( n = i \) messages,

\[
P(V) = P(R_i, R_{i-1}, ..., R_1) = \phi_i \phi_{i-1} ... \phi_1
\]

We will prove for \( n = i + 1 \),

\[
P(V) = P(R_{i+1}, R_i, R_{i-1}, ..., R_1) = \phi_{i+1} \phi_i \phi_{i-1} ... \phi_1
\]

By the definition of conditional probability we have,

\[
P(R_{i+1}, R_i, ..., R_1) = P(R_{i+1}|R_i, R_{i-1}, ..., R_1) P(R_i, R_{i-1}, ..., R_1)
\]

Since \( R_{i+1} \subseteq \text{dom}(\phi_{i+1}) \), and from Lemma 4.5.2, the variables not in \( \text{dom}(\phi_{i+1}) \) are independent of the variables in \( \text{dom}(\phi_{i+1}) \),

\[
P(R_{i+1}|R_i, R_{i-1}, ..., R_1) = P(R_{i+1} \mid \text{dom}(\phi_{i+1}) - R_{i+1})
\]

(4.5.1)

\[
= P(R_{i+1} \mid \text{dom}(\phi_{i+1}) \cap \bigcup_{j=1}^{i} \text{dom}(\phi_j))
\]

(4.5.2)

\[
= P(R_{i+1} \mid \text{dom}(\phi_{i+1}) \cap \{R_i, R_{i-1}, ..., R_1\})
\]

(4.5.3)

\[
= P(R_{i+1} \mid S_{i+1})
\]

(4.5.4)

\[
= \phi_{i+1}
\]

(4.5.5)

From Equation 4.5.5 and the inductive hypothesis,

\[
P(V) = P(R_{i+1}, R_i, ..., R_1) = P(R_{i+1} \mid R_i, R_{i-1}, ..., R_1) P(R_i, R_{i-1}, ..., R_1)
\]

\[
= \phi_{i+1} P(R_i, R_{i-1}, ..., R_1)
\]

\[
= \phi_{i+1} \phi_i \phi_{i-1} ... \phi_1
\]

\[\square\]
We have therefore shown that using the communication solution, each agent can retrieve the correct distribution over the received shared variables, and marginalize this distribution to the required input distributions. Using equations 3.2.2 or 3.2.3, the agent updates its local Bayesian network to be consistent with the received messages.
Due to the oracular assumption of AEBNs, treating published beliefs as soft evidence in subscribing agents is appropriate. Therefore, soft evidential update is a central aspect of our multiagent model. In this chapter we review several soft evidential update methods, one of which we proposed as part of our research work. It is our aim to provide insight into the advantages and disadvantages of each method to aid multiagent designers. We perform initial performance and complexity analysis of the methods reviewed.

The issue of how to deal with uncertain evidence in Bayesian networks appears in Pearl’s foundational text [46, sections 2.2.2, 2.3.3] and has recently been the subject of methodological inquiry and algorithm development (e.g., [57, 6, 29, 59, 8, 50, 45, 17]). A result of these studies has been to clarify the distinction between soft and virtual evidence. As mentioned in Section 2.2, representing uncertain probabilistic evidence as virtual evidence is appropriate when we model the reliability of an information source, while the soft evidence representation is appropriate when we want to incorporate the distribution of a variable of interest into a probabilistic model. Update based on virtual evidence (sometimes called likelihood evidence) is supported in several existing Bayesian inference engines, such as HUGIN\(^1\). This chapter is concerned with soft evidence only.

For the purpose of this dissertation, we define evidence in a Bayesian network

\(^1\)Virtual evidence is, confusingly, called soft evidence in [40].
as a collection of findings on variables of the Bayesian network. A hard finding
specifies which value (state) the variable is in. A soft finding specifies the probability
distribution of a variable. Hard evidence is a collection of hard findings. Soft evidence
is a collection of soft findings. See [57] for more general definitions of evidence. Some
authors describe the problem of update in the presence of soft evidence as a model
revision or parameter tuning problem. In the case of soft evidential update, all
evidence (hard and soft) is presented simultaneously; in the case of model revision,
the soft evidence is better considered as a constraint on the probability distribution
encoded by the model, which is modified before evidence is applied. Despite the clear
difference in the problems that are solved, similar algorithms can be used to solve
both problems, as can be seen by contrasting [50], which takes the model revision
approach, with [45], which takes the evidential update approach.

Belief update in the presence of hard evidence is carried out by conditioning.
As observed by many authors, conditioning cannot be used to update beliefs in the
presence of soft evidence. The general soft evidential update method of [57] will be
used in this paper; this general method admits several detailed algorithmic variants,
which have different efficiency characteristics with respect to network topologies and
evidence presentations. The input to the method consists of a Bayesian network
and a set of soft and hard findings. The method computes implicitly a joint prob-
ability that has two properties: (1) the evidence is respected, i.e. the findings are
marginals for the joint probability distribution; (2) the joint probability is as close as
possible to the initial distribution represented in the input Bayesian network, where
distance is measured by cross-entropy (\(I\)-divergence). The joint probability is com-
puted implicitly in that only its single-variable marginals are output. The focus of
this paper is the experimental comparison of three such variants: the big clique al-
gorithm of [57, 29], and the two wrapper-based methods of [45]. The authors of [45]
prove that the three variants compute the same distribution. The three variants are
described in the following section. We aim (in future work) to provide further insight into the appropriateness of the three variants for different network topologies and evidence presentations.

5.1 Lazy Big Clique

Valtorta et al. showed that an IPFP computation can be decomposed by proving the following lemma [57]:

**Lemma 5.1.1.** Let \( P(V) \) be a probability distribution over a set \( V \) of propositional variables. Let \( C \subseteq V \) and \( D \subseteq V \) where \( C \cup D = V \) and that for the set of all observational variables \( O_i, i = 1, ... , n \), it holds that \( \{O_1, ..., O_n\} \subseteq C \). Then \( Q^*(V) \), the \( I_1 \)-projection of \( P(V) \) on the set of all distributions having \( Q(O_i), i = 1, ..., n \) as their marginals can be computed as:

\[
Q^*(V) = Q^*_C(C) \cdot P(V \setminus C | C),
\]

where \( Q^*_C(C) \) denotes the \( I_1 \)-projection of marginal \( \sum_{V \setminus C} P(V) \) on the set of all distributions defined on \( C \) and having \( Q(O_1), ..., Q(O_n) \) as their marginals.

This is an important result that shows one can avoid performing IPFP on a full joint probability distribution. Instead, IPFP can be performed on a clique in a junction tree that contains all variables that have soft evidence. This result is exploited in the following algorithm.

The Big Clique algorithm [57] incorporates soft evidence by combining two methods: junction tree propagation and Iterative Proportional Fitting Procedure (IPFP; [12, 58, 57]). The original Big Clique algorithm modified the HUGIN propagation algorithm and therefore did not exploit d-separation properties of the underlying Bayesian network. A new version of the big clique algorithm was developed (the Lazy Big
Clique algorithm), that is more efficient by taking advantage of d-separation using the lazy propagation algorithm described in the Section 2.2.

**Algorithm 5.1.1 (Lazy Big Clique).** The lazy big clique algorithm modifies the lazy propagation algorithm as follows:

1. Construct a junction tree that includes all variables that have soft evidence in one clique - the big clique $C_1$.

2. Apply hard evidence and invoke the lazy propagation routine **Collect Evidence** on $C_1$.

3. Combine all potentials associated with $C_1$ to produce the joint probability distribution $P(C_1)$.

4. Absorb all soft evidence in $C_1$ (with the algorithm described on page 82).

5. Invoke the **Big Clique Distribute Evidence** routine. A special method is needed to distribute evidence from the big clique since during absorption of soft evidence the decomposition of potentials in $C_1$ is lost, and therefore a division by the evidence received from a neighboring clique is necessary when calculating messages to avoid passing back redundant information.

**Algorithm 5.1.2 (Big Clique Distribute Evidence).** :

1. For each clique $C_i$ adjacent to $C_1$, combine potentials of message in collect mailbox of separator $S$ between $C_i$ and $C_1$, call this result $\Phi_i^*$ - the evidence $C_1$ received from $C_i$.

2. Calculate message passed from $C_1$ to $C_i$ as follows:

$$\Phi_i^↓ = \frac{\Phi_{C_1}}{\Phi_i^*}$$

3. For each variable $X$ in $\{X \in \text{dom}(\Phi_i^↓)|X \notin S\}$
a) Marginalize out $X$.

4. Let $\Phi_i^{\downarrow *}$ be the potential obtained.

5. Store $\Phi_i^{\downarrow *}$ in the distribute mailbox of $S$.

6. Update $\Phi_{C_i} = \Phi_{C_i} \cup \Phi_i^{\downarrow *}$.

7. $C_i$ invokes the lazy propagation routine Distribute Evidence on all adjacent cliques except $C_1$.

Note that, when $\Phi_i^x = 0$, then $\Phi_{C_1} = 0$ since a 0-potential configuration cannot become a positive potential configuration during absorption of soft evidence. The junction tree algorithm has a similar property. As in the junction tree algorithm, we define $\frac{0}{0} = 1$.

**Absorption of Soft Evidence:**

We define absorption in the special big clique $C_1$ as the process by which the joint probability $P(C_1)$ is updated to satisfy the constraints imposed by soft evidence on variables $S \subseteq C_1$, where $S = \{S_1, S_2, \ldots, S_k\}$. Let $Q(C_1)$ be the joint probability after absorption. Then $\forall i \sum_{C_1 \setminus S_i} Q(C_1) = P(S_i)$, where $P(S_i)$ is the soft evidence on $S_i$, $i = 1, \ldots, k$. Absorption of soft evidence in clique $C_1$ is done using IPFP and consists of cycles of $k$ steps, one per finding. Each step corresponds to one soft finding. The procedure is as follows:

\[ Q_0(C_1) = P(C_1) \]
\[ Q_i(C_1) = \frac{Q_{i-1}(C_1) \cdot P(S_j)}{Q_{i-1}(S_j)} \]

where $j = (i - 1) \mod k + 1$. 
5.2 Wrapper Methods

Wrapper Method 1: Iterate over network

Both wrapper methods [45] utilize any existing Bayesian inferencing engine that supports virtual evidence by converting soft evidence findings into virtual evidence that are applied to the Bayesian network using standard inference. Convergence is achieved using an iterative method. For wrapper method 1, at each iteration one soft evidence finding is converted to virtual evidence and applied. The process is performed repeated until convergence as follows:

Let $P(X)$ be the joint probability of the Bayesian network $N$ obtained using standard BN inference. Let $S$ be the variables with soft evidence, where $S = \{S_1, S_2, ..., S_k\}$, and $P(S_i)$ is the soft evidence on $S_i$, $i = 1, ..., k$. This algorithm applies soft evidence by iterating over the whole network as follows:

1. $Q_0 = P(X); k = 1$

2. Repeat the following until convergence:

   a) $i = 1 + (k - 1) \mod m; j = 1 + [(k - 1)/m]$;

   b) (Convert the soft evidence to virtual evidence) Construct virtual evidence $V_{i,j}$ with likelihood ratio:

   $$L(S_i) = \frac{P(S_i)}{Q_{k-1}(S_i)}$$

   c) Obtain $Q_k(X)$ by updating $Q_{k-1}(X)$ with $V_{i,j}$ using standard BN inference.

   d) $k = k + 1$

Wrapper Method 2: Iterate over soft evidence

Wrapper method 2 is similar to the big clique algorithm in that both methods calculate the joint probability of the soft evidence variables and use IPFP to absorb soft
evidence. The big clique performs IPFP on all variables in the big clique, while the
wrapper 2 method only performs IPFP on the soft evidence variables. The wrapper 2
method converts the soft evidence to virtual evidence that is applied to the Bayesian
network using standard inference. As a result, the wrapper 2 method requires two
full propagations: one to calculate the joint probability of the soft evidence variables,
and another to calculate the posterior marginals. The process is as follows:

Let $P(X)$ be the joint probability of the Bayesian network $N$ obtained using
standard BN inference. Let $S$ be the variables with soft evidence, where $S = \{S_1, S_2, ..., S_k\}$ and $P(S_i)$ is the soft evidence on $S_i, i = 1, .., k$. Let $P(S)$ be the
joint probability of $S$. This algorithm applies soft evidence as follows:

1. Use any BN inference method on $N$ to obtain $P(S)$.

2. Absorb all soft evidence in $P(S)$ (with the algorithm described below) to obtain
   $Q(S)$.

3. (Convert the soft evidence to virtual evidence) Construct virtual evidence $V$
   with likelihood ratio:
   
   $$ L(S) = \frac{Q(S)}{P(S)} $$

4. Update the beliefs in $N$ with $V$ using standard BN inference.

Bayesian network engines of the “all-marginal” variety (junction tree based) do
not compute joint probabilities, but rather calculate single-variable marginals for all
variables. Junction tree algorithms can be modified to calculate joint probabilities
for a set of variables by adding pairwise edges between all variables of interest to
the moral graph before performing triangulation. This ensures the resulting junction
tree will contain a clique that contains all variables of interest. After propagation,
the joint probability of the variables can be constructed by combining all potentials
associated with this clique. Our implementation of the wrapper 2 method uses this
technique to calculate the joint probability of the soft evidence variables. See [4] and [26, Section 5.2] for a discussion of other methods to calculate joint probabilities in “all-marginal” algorithms.

**Absorption of Soft Evidence:**

We define absorption of soft evidence as the process by which the joint probability $P(S)$ is updated to satisfy the constraints imposed by soft evidence on variables $S$, where $S = \{S_1, S_2, ..., S_k\}$. Let $Q(S)$ be the joint probability after absorption. Then $\forall i \sum_{S \neq S_i} Q(S) = P(S_i)$, where $P(S_i)$ is the soft evidence on $S_i$, $i = 1, ..., k$. Absorption of soft evidence is done using the Iterative Proportional Fitting Procedure (IPFP) and consists of cycles of $k$ steps, one per finding. Each step corresponds to one soft finding. The procedure is as follows:

$$Q_0(S) = P(S)$$

$$Q_i(S) = \frac{Q_{i-1}(S) \cdot P(S_j)}{Q_{i-1}(S_j)}$$

where $j = (i - 1) \mod k + 1$.

5.3 Complexity Analysis

In order to discuss the complexity of each of the variants of soft evidential update, we introduce the following terms:

$n$ : the number of variables in the Bayesian network.

$r$ : the max number of states for a variable.

$p$ : the number of cliques in the junction tree.

$m$ : the max number of variables in a clique.
$n_{it}$ : the number of iterations performed.

e $e$ : the number of soft evidence variables.

From these defined constants, the complexity of each variant is easily evaluated and is shown in Table 5.1 (note all variants are based on using a junction tree for belief propagation).

<table>
<thead>
<tr>
<th>Variant</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big clique</td>
<td>$5pr^m + [2r^m + r]en_{it}$</td>
</tr>
<tr>
<td>Wrapper method 1</td>
<td>$en_{it}[5pr^m + r^m + r]$</td>
</tr>
<tr>
<td>Wrapper method 2</td>
<td>$10pr^m + 2r^m + [2r^m + r]en_{it}$</td>
</tr>
</tbody>
</table>

This result indicates that when the cost of propagation over the whole network dominates soft evidence update, then the Big Clique and Wrapper method 2 should be more efficient, while if IPFP over the Big Clique or joint distribution over the soft evidence dominates, then Wrapper method 1 is more efficient.

5.4 Experimental Setup

To evaluate the lazy big clique (referred to as big clique from here on) and wrapper algorithms, a new Bayesian reasoning engine was constructed that utilizes lazy propagation, the Bayesian Reasoning Using Soft Evidence (BRUSE) engine. BRUSE was developed using the Java framework and implements the three discussed algorithms for soft evidential update. In order to evaluate algorithm performance, an instrumentation framework was implemented into BRUSE to gather statistics during inferencing. Statistics collected are: number of table multiplication operations performed, number of table addition operations performed, number of table division operations performed, IPFP iterations required for convergence, domain size of the IPFP table, and time to perform inference.
Our testing was done on a Dell Optiplex Intel Core 2 Duo, 2.4 GHz machine with 2GB of RAM. Each test configuration was performed ten times and average statistics were calculated. The tests were performed using four Bayesian networks of varying sizes and complexity. Two of the networks were downloaded from a web-based repository [18]: stud farm (12 nodes) [26] and alarm (37 nodes) [2]. The other two networks, test71 (80 nodes) and test61 (200 nodes), were randomly generated\(^2\) to simulate complex networks. Parameters of the random generator were: max 200 nodes, max degree of 5 per node, max of 3 states per node, and max of 200 edges in the network. Table 5.2 shows statistics for the four networks. These statistics show the relative complexity of the networks and corresponding junction trees when one soft evidence finding is chosen.

Each network was tested with ten different test configurations consisting of one to ten soft evidence findings. Hard evidence was not used in our tests. Each test, randomly selects soft evidence variables accordingly to satisfy the test configuration chosen. The same set of soft evidence findings are applied to each of the three algorithms to compare their relative performance.

<table>
<thead>
<tr>
<th>Network</th>
<th>Number of Nodes</th>
<th>Number of Cliques</th>
<th>Max Clique Size</th>
<th>Triangulation Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>studfarm</td>
<td>12</td>
<td>9</td>
<td>16</td>
<td>116</td>
</tr>
<tr>
<td>alarm</td>
<td>37</td>
<td>27</td>
<td>144</td>
<td>1065</td>
</tr>
<tr>
<td>test71</td>
<td>80</td>
<td>65</td>
<td>2916</td>
<td>13793</td>
</tr>
<tr>
<td>test61</td>
<td>200</td>
<td>175</td>
<td>262144</td>
<td>347180</td>
</tr>
</tbody>
</table>

5.5 Results

We present in this section some of the experimental results obtained. Since we use the min-size heuristic, which is widely recognized as excellent [30], the cost to generate

\(^2\)Bayesian networks were generated using Fabio Cozman’s BNGenerator tool found at: http://www.pmr.poli.usp.br/ltd/Software/BNGenerator/
the junction tree is negligible. Accordingly, for all networks, we collect statistics only after the construction of the junction tree. Also, we found that, in our implementation, inference time corresponds closely to the number of elementary table operations performed, where we define number of elementary table operations as the sum of table multiplications, additions and divisions. As an example, compare Figure 5.1 with Figure 5.2. Therefore, the number of table operations provides a good measure of relative performance.

For all networks, it appears that wrapper 1 is slower than the other two methods when the number of soft evidence findings is small (less than 7 for the networks we
Figure 5.3: Number of elementary table operations for test case 3 of the alarm network.

Figure 5.4: Average number of elementary table operations for the test61 network.

consider). (We apologize to the reader for the fact that several of the graphs do not provide sufficient resolution to show this.) We conjecture that the reason is that the cost of propagation through the whole network dominates the cost of IPFP over a rather small joint probability. As a consequence, we also conjecture that this would not be the case for networks with large state spaces, for which the joint probability tables are large, even when they contain only a few nodes.

For all networks, wrapper 2 and big clique have similar run times. This is to be expected, because both methods need to compute the joint probability of the soft evidence variables, which requires, in a junction tree algorithm, the computation
of the joint probability of variables in a clique that contains all the soft evidence variables. The big clique algorithm also performs IPFP on all variables in that clique, while the wrapper 2 method only performs IPFP on the soft evidence variables. On the other hand, the wrapper 2 method uses virtual evidence, which requires two full propagations, to compute posterior marginals, while the big clique method only needs one full propagation. When the cost of propagation is higher than the cost of IPFP on the big clique, the big clique algorithm will perform better, and vice versa.

For the stud farm network (Figure 5.7), the cost of propagation in the very small junction tree is dominated by the cost of IPFP in the big clique and wrapper 2
Use of IPFP requires the computation of the joint probability of the soft evidence variable(s), by first computing the joint probability of the variables in the cliques containing the soft evidence variable(s) and then marginalizing down to the soft evidence variable(s). On the other hand, the wrapper 1 method computes posterior marginal probabilities by updating with respect to each individual soft evidence variable in turn. This computation is very fast on the small junction tree of the stud farm network. Accordingly, the wrapper 1 method is the fastest for this network.

For the alarm network (Figure 5.1), the results are similar to those for stud farm. The relatively poor performance of big clique for eight soft evidence findings is explained by a particularly difficult evidence scenario, whose performance is reported in Figure 5.3. The resulting big clique for these soft evidence findings is very large resulting in an expensive IPFP computation. A similar situation occurs for ten soft evidence findings.

For the test61 network (Figure 5.4), the state spaces of one of the cliques in the junction tree is very large, reflecting the fact that this is indeed a random network and not a typical, human-constructed, low-treewidth network [5]. The performance of the wrapper 1 algorithm is accordingly poor, because the cost of additional propagations
required by this method overcomes the savings resulting from not performing IPFP on a joint distribution. Similarly, wrapper 2 is slower than the big clique algorithm, because it performs twice the number of propagations. The spike in the number of table operations for the wrapper 1 method with nine soft evidence findings is due mainly to one difficult evidence scenario, whose performance is reported in Figure 5.5, for which the number of IPFP iterations before convergence is very high.

For the test71 network (Figure 5.6), the number of operations for the wrapper 2 method is approximately double the number for big clique. This indicates that the contribution of IPFP is negligible, while the propagation cost for probability update after IPFP dominates the number of operations. Since wrapper 2 needs to perform two such propagations, as opposed to one for the big clique algorithm, the experimental result is explained. The junction tree constructed for the wrapper one method, which does not need to include all soft evidence variables in one clique, is much simpler than the one built for the other two methods, and this explains the comparatively better performance of wrapper one for seven evidence findings.
To evaluate the proposed AEEN model, we implemented an AEEN framework using the Java SE software development kit (JDK 1.6). This implementation allows us to run agent simulations and capture various performance metrics. These performance metrics are compared with similar simulations implemented using Xiang’s MSBN framework and provide insight into the trade-offs of modeling using our AEEN model, and MSBNs. Performance metrics are collected for all agents during each phase of message passing in the agent communication graphs. In order to compare the two systems, we collect the following performance metrics:

1. Cross-entropy and CD distance w.r.t. MSBN distribution of shared variables
2. Posterior beliefs of shared variables
3. Total size of all messages sent

The first two metrics provide insight into the effect the oracular assumption has on the shared beliefs in the agent system as compared with a system that strictly adheres to d-separation properties in a centralized graphical model. Since posterior beliefs in MSBN are identical to those in a global Bayesian network model, we will compare the belief of each shared variable in our agent model to the corresponding belief in a similar simulation implemented as an MSBN. The last metric provides insight into the computational and resource implications of our model and MSBNs.

Our desire is for the multiagent simulation to be semi-realistic to reflect design issues a multiagent system designer may face designing real world systems. Our cho-
sen simulation is based on a “bio-attack” example devised by Laskey and Levitt [34]. In this example, a sophisticated coordinated multi-city bio-warfare attack is orchestrated by a terrorist organization on the United States. The terrorist organization utilizes multiple contagions to masquerade a deadly anthrax attack as a less serious cutaneous anthrax and foot-and-mouth disease outbreak in the American cattle industry. All three contagions have similar symptoms in cattle and humans.

The goal of the terrorist organization is to cause government authorities to mistakenly link illness in humans from a deadly strain of anthrax with two independent disease outbreaks in cattle. The ensuing confusion will delay detection of the terrorist plot, resulting in high civilian casualties and high economic damage.

Although the example is fictitious, it is semi-realistic due to the following facts [34]:

1. Outbreaks of foot-and-mouth disease on livestock has the potential of causing trillion dollar economic damage to the US economy.

2. Over 95% of beef processing in the United States is concentrated in a very small number of large scale factories, mainly located in large industrial cities such as Chicago, Kansas City, Denver and Dallas/Fort Worth. The animal-to-product cycle is highly efficient and it takes only a few days for the product to reach the dinner table.

3. Cutaneous anthrax can be transmitted to humans from livestock.

4. Inhalation anthrax is deadly to both humans and livestock and is easily spread in aerosol form. Only 50-100kg of weapons grade anthrax would be required to attack an urban population.

The sequence of events for the scenario are outlined in Table 6.1. The scenario proceeds from day 1 (the start of the scenario) to day 18 (the end of the scenario) for a coordinated terrorist attack. We stop at day 18 since a terrorist attack is
certainly detected due to the discovery of weapons grade inhalation anthrax in human populations. In the table, the events that are evidence the intelligence agents can gather are highlighted in bold. The goal of our simulation is to determine how well an AEBN and MSBN system can detect the terrorist attack. Note that the word agent in the phrase “terrorist agents” in Table 6.1 does not refer to an intelligent computational agent.

<table>
<thead>
<tr>
<th>Day</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day 1</td>
<td>Terrorist agents infect Chicago cattle herds at target stockyards with cutaneous anthrax.</td>
</tr>
<tr>
<td>Day 3</td>
<td>Terrorist agents infect Chicago cattle herds with foot-and-mouth disease.</td>
</tr>
<tr>
<td>Day 5</td>
<td><strong>First reports of anthrax and foot-and-mouth symptoms in Chicago cattle herds.</strong> Terrorist agents spray Chicago herds with inhalation anthrax. Simultaneously, terrorist agents infect Kansas City cattle herds at target stockyards with cutaneous anthrax.</td>
</tr>
<tr>
<td>Day 7</td>
<td>Terrorist agents use crop duster to spray Chicago with inhalation anthrax. Simultaneously, terrorist agents infect Kansas city cattle herds with foot-and-mouth disease.</td>
</tr>
<tr>
<td>Day 8</td>
<td><strong>Lab tests confirm cutaneous anthrax at Chicago stockyard.</strong></td>
</tr>
<tr>
<td>Day 9</td>
<td>Terrorist agents spray Kansas city cattle herds with inhalation anthrax. Simultaneously, terrorist agents infect Denver cattle herds at target stockyards with cutaneous anthrax.</td>
</tr>
<tr>
<td>Day 11</td>
<td>Terrorist agents use crop duster to spray Kansas city with inhalation anthrax.</td>
</tr>
<tr>
<td>Day 12</td>
<td><strong>Lab tests confirm inhalation anthrax at Chicago stockyard.</strong> Terrorist agents infect Denver cattle herds with foot-and-mouth disease.</td>
</tr>
<tr>
<td>Day 13</td>
<td><strong>Lab tests confirm foot-and-mouth disease at Chicago stockyard.</strong> Terrorist agents spray Denver herds with inhalation anthrax.</td>
</tr>
<tr>
<td>Day 14</td>
<td><strong>Lab tests confirm cutaneous anthrax at Kansas city stockyard.</strong></td>
</tr>
<tr>
<td>Day 15</td>
<td><strong>Lab tests confirm cutaneous anthrax at Denver stockyard.</strong> Terrorist agents use crop duster to spray Denver with inhalation anthrax.</td>
</tr>
<tr>
<td>Day 16</td>
<td><strong>Lab tests confirm inhalation anthrax at Kansas city stockyard.</strong></td>
</tr>
<tr>
<td>Day 17</td>
<td><strong>Lab tests confirm inhalation anthrax at Denver city stockyard.</strong></td>
</tr>
<tr>
<td>Day 18</td>
<td><strong>Lab tests confirm inhalation anthrax in human populations in Chicago.</strong></td>
</tr>
</tbody>
</table>

To detect and reason about the scenario, we implement a fictitious distributed detection network that attempts to detect the unfolding scenario and minimize damage
from the terrorist plot. Laskey and Levitt proposed a single Bayesian network\(^1\) to illustrate the power of MEBNs, we constructed a similar Bayesian network (Figure 6.1) for reasoning about the scenario. We will use this model as a guide for constructing two multiagent systems: one based on our AEBN model, and the other based on an MSBN.

Our simulation includes early detection agents representing agents located in meat processing facilities, and governmental monitoring agencies. Information gathered from early detection agents is reported to local threat assessment agents as part of a nationalized monitoring and detection network. Reports from local threat assessment agents are transmitted to a national incident agent which is responsible for assessing attack types detected and issue alerts to appropriate authorities of the probability of a coordinated terrorist attack. Figure 6.2 shows a summarized overview of the proposed agent communication graph for the AEBN simulation (only Chicago and Kansas

\(^{1}\)Constructed using Multi-Entity Bayesian Networks (MEBN).
agents are shown). The full multiagent system contains seventeen agents: one incident agent, eight attack type agents and eight early indicator agents. The attack type and early indicator agents are divided by region, where each region has two attack type agents and two early indicator agents for human and livestock population monitoring and testing. In our simulation, there are four regions: Chicago, Kansas, Denver and Dallas. The agent set associated with each region are essentially identical, except variable labels are specific to each particular region. A similar agent decomposition is devised for an MEBN multiagent system as discussed in Section 6.2.

![Communication graph for bio-attack AEBN simulation (only Chicago and Kansas agents shown).](image)

**Figure 6.2:** Communication graph for bio-attack AEBN simulation (only Chicago and Kansas agents shown).

**Early Indicator Agents**

Early indicator agents represent early detection report agents that monitor abnormal rates of illness or deaths in human and livestock populations and calculate their belief the observations indicate possible anthrax or foot-and-mouth disease. The joint probability of early indicators is calculated and passed to the appropriate attack type agent.
Attack Type Agents

Attack type agents represent early government monitoring and testing facilities that receive early detection reports from early indicator agents and also are capable of performing tests for specific strains of anthrax and foot-and-mouth disease. Each attack type agent computes its belief that an inhalation, cutaneous or foot-and-mouth disease outbreak in the target population has occurred given all the available evidence. The joint probability of the outbreak is calculated and passed to the incident agent.

Incident Agent

The incident agent represents a national alert agent that receives reports of outbreaks from attack type agents and assesses the probability of a terrorist attack and characterizes the type of terrorist attack. The incident agent relies on the reports from the attack type agents and fuses the information into its internal model. We envision in a real world scenario the incident agent would initiate alerts to appropriate authorities if the belief of a terrorist attack reached an appropriate threshold.

6.1 AEBN Multiagent Simulation

In our AEBN multiagent simulation, each agent calculates and processes messages according to the following:

1. Messages are only sent to subscribers when new evidence is discovered
2. Messages are computed using current beliefs based on all available evidence
3. Replace previous evidence received with new evidence received
4. The most recent evidence is used to reason

Each agent in an AEBN has an internal Bayesian network used for reasoning given local and external evidence received. The Bayesian network of the incident agent is
shown in Figure 6.3. An example of the Bayesian networks for the attack type agents is shown in Figure 6.4 and Figure 6.6 for the Chicago human and livestock attack type agents respectively. Finally, an example of the Bayesian networks for the early indicator agents is shown in Figure 6.5 and Figure 6.7 for the Chicago human and livestock early indicator agents respectively.

In the Bayesian network figures, the shaded nodes with dashed borders represent observation nodes (as defined in Section 3.2) that are introduced to absorb the messages from publishing agents, and nodes that are shaded with double lined borders represent variables that can be subscribed to by other agents.

For the purposes of our simulation, we assume perfect communication in the agent system. Each agent first receives all messages from publishing agents it is subscribed to then performs belief revision. After new beliefs have been computed, each agent computes and sends messages to subscribing agents. This process is performed for each evidence phase described in Section 6.3.

### 6.2 MSBN Multiagent Simulation

The MSBN multiagent simulation is constructed using a similar decomposition of the global Bayesian network as the AEBN simulation. We used Xiang’s publicly available WEBWEAVER-III toolkit to construct and validate the soundness of sectioning (see Section 3.1 for details) of the MSBN. Figure 6.20 shows a summarized version of the resulting linked junction forest for the MSBN. Each junction tree in the linked junction forest represents an agent. The agent roles are defined similarly to our AEBN decomposition and comprises the same set of seventeen agents.

Belief propagation in an MSBN is analogous to propagation in a junction tree.

---

2Perfect communication means no latency, transmission failures or corrupted messages occur in the network.

3WEBWEAVER-III is available for download at http://www.cis.uoguelph.ca/~yxiang/
where branches of the junction tree are distributed to agents, rather than centralized in one agent. Therefore, as opposed to our AEBN simulation, where agents only transmit messages when new evidence has arrived, in MSBN, messages are transmitted in both directions for each agent during each message passing phase similar to the collect and distribute evidence phases in a junction tree propagation algorithm.
A full message passing phase is initiated when an agent receives new evidence and revises its beliefs. The evidence is propagated throughout the agent system so all agents are consistent over their shared beliefs given the new evidence.

In our MSBN simulation, each message passing phase corresponds to one agent receiving evidence. The evidence phases are described in Section 6.3. As with the AEEN simulation, we assume perfect communication in the MSBN simulation.
6.3 Evidence Phases

From the sequence of events defined in Table 6.1, the evidence phases defined in Table 6.2 will be used for the simulations. Each phase is defined as evidence for that phase being entered in the appropriate agent, which initiates message passing and belief update on the agent system according to the semantics of each simulation.

Table 6.2: Evidence phases for Bio-attack simulation.

<table>
<thead>
<tr>
<th>Phase 1</th>
<th>Initial state (no evidence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 2</td>
<td>Sick cows observed in Chicago</td>
</tr>
<tr>
<td>Phase 3</td>
<td>Positive test for cutaneous anthrax in Chicago livestock</td>
</tr>
<tr>
<td>Phase 4</td>
<td>Positive test for inhalation anthrax in Chicago livestock</td>
</tr>
<tr>
<td>Phase 5</td>
<td>Positive test for foot-and-mouth disease in Chicago livestock</td>
</tr>
<tr>
<td>Phase 6</td>
<td>Positive test for cutaneous anthrax in Kansas city livestock</td>
</tr>
<tr>
<td>Phase 7</td>
<td>Positive test for cutaneous anthrax in Denver livestock</td>
</tr>
<tr>
<td>Phase 8</td>
<td>Positive test for inhalation anthrax in Kansas city livestock</td>
</tr>
<tr>
<td>Phase 9</td>
<td>Positive test for inhalation anthrax in Denver livestock</td>
</tr>
<tr>
<td>Phase 10</td>
<td>Positive test for inhalation anthrax in Chicago human population</td>
</tr>
</tbody>
</table>

Since the goal of the simulations is the detection and classification of the terrorist attack we will focus on comparing each simulation’s beliefs of the BioAttackType variable, which has states: Coordinated Bio Attack, Local Bio Attack, Non-Bio Attack, and No Attack.

6.4 Enhanced AEBN model

During the course of our experimentation, we discovered a modeling issue with our originally defined AEBN simulation. External evidence received by attack type agents had too strong an influence over local evidence. We identified this as a general modeling issue with AEBN systems and this behavior may not be appropriate in some situations. For example, when an external observer agent reports their unreliable belief of the presence of a disease to a subscriber agent which can obtain local evidence in the form of a test that can confirm or deny the existence of the disease with a high
accuracy, the observation received is not as important as the local evidence.

In general, we can state the problem as: the reliability or importance of the external evidence needs to be modeled so it is offset or discounted by more reliable or important evidence the local agent acquires.

The situation in our simulation is similar, where our early indicator agents make observations that suggest (or indicate) the presence of anthrax or foot-and-mouth disease, and the attack type agents can perform a highly accurate test to confirm or deny the presence of the contagions.

To account for these type of situations we propose the following modeling technique in AE VN systems:

1. Introduce a mediating variable in the subscriber agent

2. The mediating variable acts as a “switch” that turns off or discounts the affect of the external evidence when more accurate local evidence is present

Figure 6.8 shows the general modeling technique, and the mediating variable conditional probability table is shown in Table 6.3. In our implementation, the external evidence is ignored (using a uniform distribution) if a test has been performed, but more generally it could be discounted using any suitable distribution. In practice, either the discount factor could be specified by a designer or an agent could maintain a discount factor based on the historic reliability of the communicating agent.

<table>
<thead>
<tr>
<th>Test</th>
<th>false</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disease</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>true</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

In Figures 6.9 and 6.10 we show the revised Bayesian network models of the Chicago human and livestock attack type agents. The Kansas, Denver and Dallas attack type agents are modified similarly.
Figure 6.8: Discounting external evidence given an accurate test modeling technique.

We will refer to this revised model as the Enhanced AEBN simulation or simply AEBN v2, and the first AEBN simulation as original AEBN simulation or AEBN v1.

Figure 6.9: Revised Bayesian network for Chicago Human Attack type agent.

Figure 6.10: Revised Bayesian network for Chicago Livestock Attack type agent.
6.5 Simulation Results

In this section we present the results of our simulations. As discussed in the beginning of this chapter, several performance metrics were collected to compare the predictive ability and efficiency of the simulations.

Tables 6.4, 6.5, 6.6 and 6.7 show the belief of the states of the BioAttackType variable in the three simulations (MSBN, AEBN v1 and AEBN v2) as the evidence phases progress. Figures 6.11, 6.12, 6.13 and 6.14 show the corresponding plots. From these plots we see AEBN v2 performs closer to the MSBN simulation than AEBN v1. Both AEBN simulations respond similarly to the MSBN simulation, but are not as sensitive to the evidence presented in the evidence phases.

The MSBN detects the coordinated terrorist attack at phase 5 with a belief of 63.262%, while AEBN v1 detects the coordinated terrorist attack at phase 9 with a belief of 71.290% and AEBN v2 detects the coordinated terrorist attack one phase earlier at phase 8 with a belief of 68.463%. The superior performance of MSBN can be accounted for by the loss of some dependence relationships in the AEBN models due to the Oracular assumption. We draw the analogy of AEBN being like a naive Bayes model and MSBN being like a general bayes model. However, AEBN is far more powerful than a naive Bayes model since it does not sacrifice all dependence relationships.

We note that in a realistic setting, an elevated risk of a terrorist attack of even a modest amount would trigger a national terrorist alert. If we applied such an approach in the incident agent with an alert threshold of 10% an MSBN would detect an unfolding terrorist attack one phase earlier at phase 4 and our AEBN model at phase 5 for our enhanced model and at phase 6 for our original AEBN model. This earlier detection could mitigate some of the damage to civilians and livestock in Kansas and Denver.

To determine a principled measure of the difference of the distributions of the
Table 6.4: Beliefs of Coordinated Bio Attack over scenario phases.

<table>
<thead>
<tr>
<th></th>
<th>MSBN - CoordAttck</th>
<th>AEBN v1 - CoordAttck</th>
<th>AEBN v2 - CoordAttck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>0.01</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td>Phase 2</td>
<td>0.014</td>
<td>0.012</td>
<td>0.011</td>
</tr>
<tr>
<td>Phase 3</td>
<td>0.148</td>
<td>0.071</td>
<td>0.086</td>
</tr>
<tr>
<td>Phase 4</td>
<td>8.373</td>
<td>0.912</td>
<td>1.452</td>
</tr>
<tr>
<td>Phase 5</td>
<td>63.262</td>
<td>4.939</td>
<td>10.430</td>
</tr>
<tr>
<td>Phase 6</td>
<td>90.241</td>
<td>18.734</td>
<td>39.809</td>
</tr>
<tr>
<td>Phase 7</td>
<td>93.508</td>
<td>27.068</td>
<td>48.031</td>
</tr>
<tr>
<td>Phase 8</td>
<td>95.181</td>
<td>49.479</td>
<td>68.463</td>
</tr>
<tr>
<td>Phase 9</td>
<td>95.971</td>
<td>71.290</td>
<td>81.353</td>
</tr>
<tr>
<td>Phase 10</td>
<td>97.829</td>
<td>78.172</td>
<td>86.828</td>
</tr>
</tbody>
</table>

Table 6.5: Beliefs of Local Bio Attack over scenario phases.

<table>
<thead>
<tr>
<th></th>
<th>MSBN - LocalAttck</th>
<th>AEBN v1 - LocalAttck</th>
<th>AEBN v2 - LocalAttck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>0.040</td>
<td>0.040</td>
<td>0.040</td>
</tr>
<tr>
<td>Phase 2</td>
<td>0.059</td>
<td>0.056</td>
<td>0.049</td>
</tr>
<tr>
<td>Phase 3</td>
<td>0.337</td>
<td>0.237</td>
<td>0.264</td>
</tr>
<tr>
<td>Phase 4</td>
<td>3.573</td>
<td>1.327</td>
<td>2.014</td>
</tr>
<tr>
<td>Phase 5</td>
<td>9.760</td>
<td>3.677</td>
<td>6.983</td>
</tr>
<tr>
<td>Phase 6</td>
<td>8.417</td>
<td>13.475</td>
<td>22.443</td>
</tr>
<tr>
<td>Phase 7</td>
<td>6.152</td>
<td>19.184</td>
<td>26.154</td>
</tr>
<tr>
<td>Phase 8</td>
<td>4.803</td>
<td>25.234</td>
<td>22.044</td>
</tr>
<tr>
<td>Phase 9</td>
<td>4.025</td>
<td>17.673</td>
<td>12.296</td>
</tr>
<tr>
<td>Phase 10</td>
<td>2.169</td>
<td>16.770</td>
<td>9.418</td>
</tr>
</tbody>
</table>

BioAttackType variable of the simulations, we calculated the CD-distance and $I$-divergence\(^4\) of MSBN and AEBN v2. Only AEBN v2 was compared to MSBN since the plots in Figure 6.11-6.14 indicate it is overall superior to AEBN v1. The resulting distance measure results are shown in Figure 6.15 plotted over the evidence phases.

The $I$-divergence is largest during phase 5, which matches our intuition by inspecting the plots of belief change that show the largest change in belief in the MSBN simulation for Coordinate Bio Attack as $\Delta 55$ and No Attack as $\Delta 61$, while in the AEBN v2 simulation the change is $\Delta 4$ and $\Delta 6.4$ respectively. Overall, the $I$-divergence between the two simulations is not very large with the highest value being 1.257 during

\(^4\)Both CD-distance and $I$-divergence were calculated using $\log_{10}$ rather than $\ln$ or $\log_{10}$. 

106
Table 6.6: Beliefs of Non-Bio Attack over scenario phases.

<table>
<thead>
<tr>
<th></th>
<th>MSBN - NonBioAttck</th>
<th>AEBN v1 - NonBioAttck</th>
<th>AEBN v2 - NonBioAttck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>0.1500000</td>
<td>0.1500000</td>
<td>0.1500000</td>
</tr>
<tr>
<td>Phase 2</td>
<td>0.1500000</td>
<td>0.1502337</td>
<td>0.1501284</td>
</tr>
<tr>
<td>Phase 3</td>
<td>0.1510000</td>
<td>0.1515000</td>
<td>0.1515951</td>
</tr>
<tr>
<td>Phase 4</td>
<td>0.1380000</td>
<td>0.1527319</td>
<td>0.1540521</td>
</tr>
<tr>
<td>Phase 5</td>
<td>0.0620000</td>
<td>0.1661524</td>
<td>0.1773863</td>
</tr>
<tr>
<td>Phase 6</td>
<td>0.0040000</td>
<td>0.1156992</td>
<td>0.0745359</td>
</tr>
<tr>
<td>Phase 7</td>
<td>0.0010000</td>
<td>0.0910298</td>
<td>0.0499422</td>
</tr>
<tr>
<td>Phase 8</td>
<td>0.0008840</td>
<td>0.0436203</td>
<td>0.0169584</td>
</tr>
<tr>
<td>Phase 9</td>
<td>0.0004119</td>
<td>0.0186277</td>
<td>0.0110681</td>
</tr>
<tr>
<td>Phase 10</td>
<td>0.0002104</td>
<td>0.0142564</td>
<td>0.0098572</td>
</tr>
</tbody>
</table>

Table 6.7: Beliefs of No Attack over scenario phases.

<table>
<thead>
<tr>
<th></th>
<th>MSBN - NoAttck</th>
<th>AEBN v1 - NoAttck</th>
<th>AEBN v2 - NoAttck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>99.8</td>
<td>99.800</td>
<td>99.800</td>
</tr>
<tr>
<td>Phase 2</td>
<td>99.777</td>
<td>99.782</td>
<td>99.789</td>
</tr>
<tr>
<td>Phase 3</td>
<td>99.364</td>
<td>99.540</td>
<td>99.498</td>
</tr>
<tr>
<td>Phase 4</td>
<td>87.917</td>
<td>97.608</td>
<td>96.380</td>
</tr>
<tr>
<td>Phase 5</td>
<td>26.916</td>
<td>91.218</td>
<td>82.409</td>
</tr>
<tr>
<td>Phase 6</td>
<td>1.338</td>
<td>67.676</td>
<td>37.673</td>
</tr>
<tr>
<td>Phase 7</td>
<td>0.339</td>
<td>53.657</td>
<td>25.765</td>
</tr>
<tr>
<td>Phase 8</td>
<td>0.016</td>
<td>25.244</td>
<td>9.475</td>
</tr>
<tr>
<td>Phase 9</td>
<td>0.004</td>
<td>11.019</td>
<td>6.341</td>
</tr>
<tr>
<td>Phase 10</td>
<td>0.001</td>
<td>5.044</td>
<td>3.744</td>
</tr>
</tbody>
</table>

phase 5 and all other phases being less than 1. This indicates that AEBN v2 performs closely to MSBN overall.

The CD-distance results differ dramatically from $I$-divergence and identify a weakness in using this measure as a distance between distributions. CD-distance captures the worst case distance between two distributions where as $I$-divergence is a weighted average. We feel the weighted average is more representative and explains why $I$-divergence is a popular metric for comparing two distributions. As discussed above, one would expect the distance to be greatest during phase 5, rather than phase 10 as indicated by CD-distance. In our comparison, CD-distance of the two distributions is a monotonically increasing function, which is counter intuitive. CD-distance does not
weight the distance by the likelihood of events as does $I$-divergence which we feel is a strong weakness of this measure and limits its applicability for providing an accurate measure of the variability between two distributions. However, CD-distance has some nice properties such as being a true distance measure and bounding the difference of beliefs captured by two probability distributions. For our purposes, these properties
are not needed since we are comparing distance of our AEBN simulation to MSBN which we treat as a “gold standard”, hence symmetry of our distance measure is not needed, nor is bounding of belief difference.

Finally, Figure 6.16 shows the communication cost of AEBN and MSBN over the scenario phases. Over all evidence phases, the AEBN has lower communication cost
due to messages only in one direction: from publisher to subscriber. Additionally, agents only send messages when they have revised their beliefs which can result in considerable savings in communication cost as can be seen in Phases 2 to 10. Conversely, the MSBN simulation communication cost is constant because each evidence phase corresponds to propagating evidence in both directions over the links in the link tree to maintain global consistency.

For large agent networks where evidence is seldom received by a subset of agents, we posit an AEBN system can have significantly better communication performance over an MSBN system, provided the network graph is sparsely connected. In the next section we analyze the scalability of AEBN systems in terms of the communication cost versus network size and number of network links.

6.6 Scalability of AEBN Multiagent Systems

To evaluate the scalability of AEBN systems we first perform worst case communication analysis. Consider an AEBN system of \( n \) agents, which are ordered and labelled according to each agent’s index in the order. The worst case communication graph corresponds to a fully connected DAG. We create this communication graph as follows:

1. Let \( G \) be a communication graph over \( n \) agents. Let \( O \) be an ordering of the
agents, where $i$ refers to the $i$th agent in $O$.

2. For each agent $i$ in the ordering $O$
   a) Add a directed edge from $i$ to all agents that follow $i$ in the order $O$.

In this communication graph each agent will have $i - 1$ incoming edges and $n - i$ outgoing edges. For simplicity, let us assume each agent sends a message of size 2 along each of its outgoing edges (note the size of the message is not important since we are only concerned with an asymptotic upper bound on the communication complexity). Further, since multiply connected nodes exist in the graph, rumors are present in the communication graph. Each agent $i$ has a directed edge from all agents with a label $< i$. Each of these agents, $j$, will carry redundant information from agents with a label $< j$. Therefore, any AEBN system with this communication graph will require the communication solution to avoid the influence of redundant information and expand the edge labels accordingly. Figure 6.17 depicts a graphical representation of the resulting redundancy graph.

The following summation calculates the cost of expanded communication in this redundancy graph:

Figure 6.16: Total communication cost of AEBN v2 and MSBN over scenario phases.
\[ \sum_{i=1}^{n-1} (n - i) * 2^i \]

Which is bounded above by:

\[ \sum_{i=1}^{n-1} (n - i) * 2^i < n * n * 2^n = O(n^2 2^n) \]

This result shows the communication complexity in an AEBN system is intractable for densely connected communication graphs. If the communication is sparsely connected and undirected cycles are minimized, communication in AEBN systems is tractable which we evaluate experimentally.

To analyze the impact of the communication graph density on an AEBN system’s communication complexity, we generated random communication graphs where we controlled the number of agents in the graph and number of edges. We first generated a random spanning tree over the agents to ensure the graph is not disconnected and randomly added additional edges, ensuring that the resulting graph was a DAG.

Let \( N \) be the number of agents in the graph, then the initial spanning tree has \( N - 1 \) edges. This agent graph represents the minimum agent communication graph and is the lower bound on communication cost where communication cost is linear in the number of agents: \( 2 * (N - 1) \). To explore the effect graph density has on the communication cost, we generated five sets of graphs where the number of edges

Figure 6.17: Theoretical scalability of AEBN.
were fixed at: \((N - 1), N, 1.25N, 1.5N \text{ and } 2N\). For each set, we generated graphs with six different fixed number of agents: 5, 10, 20, 40, 80 and 160. Since there is much variability in the topology of the generated graphs, 20 graphs of each configuration were generated and average communication costs calculated. The results of the generated graphs is shown in Figure 6.18.

![Figure 6.18: Scalability of AEBN as number of agents and edges increases.](image)

These results indicate that when the number of edges in the graph is \(1.25 \times N\) or below, the average cost of communication is linear or near linear in the number of agents, when the size of the network is below 160 agents. As the number of agents increases the cost grows exponentially. When the number of edges is \(1.5 \times N\) or greater, the cost grows rapidly for networks greater than 40 agents. If we assume floating point numbers are represented with 64 bits, the highest average cost is \(\frac{2609853163 \times 64}{1024} = 163,115,823KB\) when a network has 160 agents and 320 edges. This communication cost is highly impractical with current networking technology. The next highest communication cost is \(349,767KB\) when the network has 160 agents and 240 edges,
which is more practical, but still too high for a large class of networks. When there are 160 agents with 200 edges, the cost is only 258KB which is highly practical. These experimental results indicate that when the communication graph is sparse ($\leq 1.25 \times N$ edges) an AEBN system scales fairly well.

6.7 Extended Simulation

To illustrate some of the flexibility inherent in AEBNs that is not present in MSBNs, consider an extended simulation where we wish to incorporate information from the Center for Disease Control (CDC). The CDC tracks outbreaks of diseases and is responsible for assessing and controlling risks to US citizens. Figure 6.19 depicts an extended agent communication graph where a CDC threat agent communicates risk assessment of disease outbreaks to the human early indicator agents. The early indicator agents incorporate this information to adjust their sensitivity of monitoring for early indications of disease in human populations. The resulting extended communication graph is no longer a tree and has multiple node disjoint paths between the incident agent and CDC threat agent. The node disjoint paths introduce rumors into the AEBN system which can be removed using the communication solution defined in Section 4.3.

The extended communication graph is not possible for an MSBN system, since the agent graph must be a tree. In order to incorporate the CDC threat agent, the agent communication graph would need to be reorganized into a less natural or undesirable communication graph. Physical restrictions of the distributed nature of the system may make this restructuring costly, inefficient or even impossible. It may be necessary to completely rebuild the MSBN to ensure soundness of sectioning is maintained and that a valid link tree is created for agent communication. Without a global Bayesian network model as a starting point for creating the MSBN system, the difficulty of incorporating new agents would increase dramatically.
Figure 6.19: Communication graph for expanded bio-attack AEBN simulation.
Figure 6.20: Linked Junction Forest for bio-attack MSBN simulation (only Chicago and Kansas agents shown).
CHAPTER 7

CONTRIBUTIONS AND FUTURE WORK

A central goal of this research was to allow easier design of probability-based agents and multiagent systems, resulting in rational decision making. A multiagent framework was presented and compared with other proposed frameworks where advantages and disadvantages of each are outlined. A central problem of message passing in probabilistic systems is the familiar rumor problem, where cycles in message passing cause redundant influence of beliefs. We develop algorithms to identify and solve the rumor problem in the context of our multiagent system. Central to our message passing scheme is the notion of soft evidential update. Traditional propagation algorithms are not compatible with soft evidence. We propose a new propagation algorithm that is based on Lazy propagation and compare the theoretical and experimental performance with other proposed solutions. To evaluate our multiagent model, we devised a simulation that we implemented as an AEBN and MSBN to compare quantitatively the two formalisms. Finally, we analyzed the scalability of our agent model.

7.1 CONTRIBUTIONS

The following outlines the main contributions of our work:

1. We proposed and formalized a probabilistic multiagent system.

2. We proposed a communication solution to solve the rumor problem and have
proven its correctness under a coherence assumption.

3. We characterized the problem of coherence and proposed a design restriction and runtime solution to ensure global coherence.

4. We qualitatively compared our formalism with a prominent probabilistic multiagent system, MSBNs.

5. We have proposed a soft evidential update method for revising agent beliefs and performed an evaluation of our methods theoretical and experimental performance with other proposed solutions.

6. We devised a multiagent simulation for quantitatively comparing our agent model to MSBNs.

7. We evaluated the theoretical and experimental scalability of our multiagent system.

7.2 Future Work

During the course of our research we have identified several possible avenues for further research:

1. Investigate AEBN communication optimizations.

   a) To lower communication costs, the communication graph can be analyzed and redundant communication links could be removed. This situation can occur in the redundancy graph, where expanded messages render some message passing unnecessary.

   b) The passing of large joint probability tables between agents is very expensive, and it may be possible to decompose the messages into a factorized
representation that requires far less communication overhead during message passing.

2. Enhancements to soft evidential update to better take advantage of d-separation during IPFP and increase performance.

   a) IPFP over a joint probability distribution is an expensive operation. If variables in the distribution are d-separated, the distribution could be decomposed and IPFP could be performed individually on each factor to dramatically lower computational costs of belief revision.

3. Since soft evidential update is a central aspect of our probabilistic multiagent model, further performance evaluation of soft evidential update methods is necessary to assist agent designers in choosing the appropriate update method to use. In particular, the advantages and disadvantages of each method should be characterized based on the structure of the graphical probabilistic models.

4. Implementation issues of AEBNs should be explored such as dynamic multiagent networks, handling of communication failures, and resolving inconsistent or conflicting evidence.

5. Characterize the joint probability distribution of shared variables represented by an AEBN system.

   a) In our research, we proved each agent can remove redundant information from received messages using the communication solution. However, proving these beliefs are consistent with a joint probability distribution that is compactly represented by the combined AEBNs is challenging due to the asymmetric nature of the Oracular assumption. Recent research in identifiability in causal Bayesian networks, such as [20, 19] shows promise in representing a joint probability distribution with asymmetric constraints.
These recent results should be explored further to prove stronger properties of AEBNs and solving the rumor problem.
BIBLIOGRAPHY


